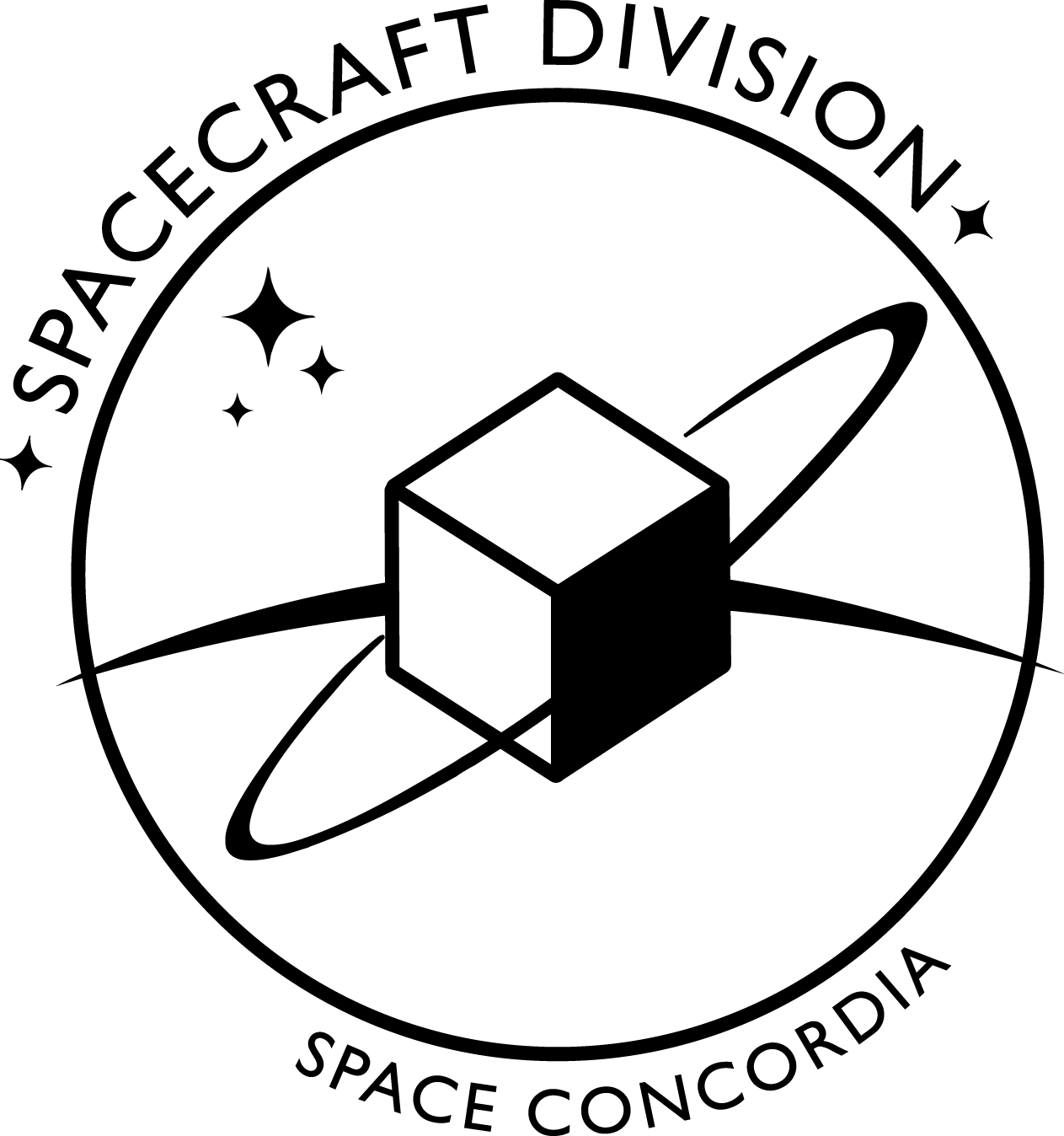
SCSD-LP-EPS-002-B

Peripherals Design



Prepared by

[Gabriel Dubé](mailto:gabriel.dube@spaceconcordia.ca)

Electrical and Power System Lead

Jan 26, 2023

# **Revision History**

| **Date** | **Revision** | **Changes** |
| --- | --- | --- |
| 1-Feb-2023 | A | Initial Release |
| -May-2023 | B | Added DACs, IOExpanders, Magnetometer, DSS and RADFET |

# **Table of Contents**

[**Revision History 2**](#_ad5thlvqzfhu)

[**Table of Contents 3**](#_z2p7d9s9al0w)

[**References 4**](#_ublz6vn9mh8u)

[**Abbreviations and Definitions 5**](#_z1bndbk4cgwt)

[**Introduction 7**](#_meu3f6nbbwk8)

[**Peripherals 8**](#_3kwz5opnz7s6)

[ADC 8](#_z2z2eratjz2m)

[DAC 14](#_ouenbu52unt4)

[RTC 17](#_o52hka1wow2m)

[I/O Expanders 20](#_na3mii1jhb2f)

[Transceivers 23](#_s4fw1xy4zrfm)

[Flash Storage 24](#_kzt8hsb6z8b7)

[Random Access Memory (RAM) 26](#_zg17qmnx7fo)

[SD Card Reader 28](#_6zgrrsnh97ce)

[Magnetometer 29](#_tkcrgwbb2fcj)

[Digital Sun Sensor 31](#_d58vjqhaj4ym)

[RADFET 32](#_pgjlvqscyi)

[Push Buttons 32](#_6kwvb0rtp51f)

# 

# **References**

[1] CUBICS - ELEC - Learning Resources - Learning Package [internal document on Google Drive], “Serial Communication”. Available from: <https://docs.google.com/document/d/1fTaHIaRPVIfN-HuLPqiPyxERU_JW75ML_xxsHw8t4RE/edit?usp=sharing>

[2] CUBICS - ELEC - Learning Resources - Learning Package [internal document on Google Drive], “MCU Design”. Available from: <https://docs.google.com/document/d/1C6u4hayMrO_3dPMdv_w5Uh5XeICUirQNjUL7YavPuvA/edit?usp=share_link>

[3] Arduino Documentation - Built-in example [website], “Debouncing on a Pushbutton”. Website link: <https://docs.arduino.cc/built-in-examples/digital/Debounce>

[4] CircuitMess Teaching/Learning Material [website], “Understanding the SD Card Slot”. Website link: <https://community.circuitmess.com/t/understanding-the-sd-card-slot/2801>

[5] CUBICS - ELEC - Learning Resources - Learning Package [internal document on Google Drive], “Power”. Available from: <https://docs.google.com/document/d/1pKWWyXrTBQWnoESd13gtv7d2UHne3ekIcaZ94zOk2eo/edit?usp=sharing>

[6] Texas Instruments [PDF, Datasheet], “TPS768xx-EP: Fast-Transient-Response 1-A Low-Dropout Voltage Regulators datasheet (Rev. B)”. Available at:

<https://www.ti.com/lit/ds/symlink/tps76801-ep.pdf?ts=1678378637738&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTPS76801-EP>

# **Abbreviations and Definitions**

| **Terminology** | **Definition** |
| --- | --- |
| ADC | Analog to Digital Converter |
| ADCS | Attitude Determination and Control System |
| CAN | Controller Area Network |
| CDH | Command and Data Handling |
| CM | Common Mode |
| CSA | Canadian Space Agency |
| DM | Differential Mode |
| DNP | Do Not Place |
| ECU | Electrical Control Unit |
| ESD | Electrostatic Discharge |
| GPIO | General Purpose Input/Output |
| IC | Integrated Circuit |
| LDO | Low Drop-Out |
| LED | Light Emitting Diode |
| LVDS | Low-Voltage Differential Signaling |
| MCU | Micro Controller Unit |
| PDS | Power Distribution System |
| PSS | Power Supply System |
| RTC | Real Time Clock |
| SC-FREYR | Fermentation R- Extraterrestrial Yeast R- |
| SPEAR-M7 | Flight computer of SC-ODIN |
| SPI | Serial Peripheral Interface |
| TVS | Transient voltage suppressor |
| UART | Universal Asynchronous Receiver-Transmitter |
| UART | Universal Synchronous and Asynchronous Receiver-Transmitter |
| USB | Universal Serial Bus |

# 

# **Introduction**

Peripherals refer to all the external devices used in digital designs. Most of the time, these allow to implement functionalities that MCUs typically do not have integrated. Great examples are protocol transceiver, real time clocks, DRAM and much more. This document will do an overview of typical implementations of these peripheral devices, but it is important to keep in mind that these designs will change with specific ICs.

# **Peripherals**

## ADC

How does it work

Using an analog to digital converter is one of the easiest ways to implement different types of sensing, which makes this peripheral very relevant for designs that need to regulate themselves. Electrical systems that will operate in space need a lot of self regulating, so using ADCs in their design is important.

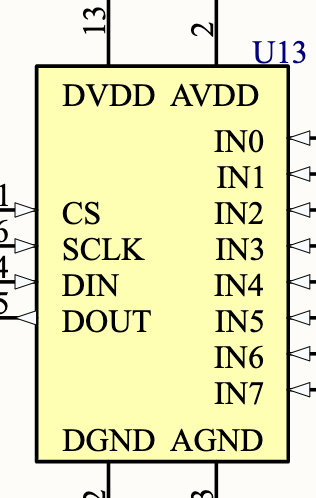
There are two main implementations of ADCs used in our CubeSat platform: MCU internal ADC and external ADC. As much as the first implementation sounds like the easiest and better way to do it, given most MCU have internal ADCs that can be used, for space application it is not always the better option. Digital chips are usually more sensitive to radiation, and relying on an MCU to both operate the system and do the self regulation is asking for a single point of failure. For this reason, external ADCs were widely used in SC-ODIN and will be used in SC-FREYR.

The idea behind an ADC is quite simple: it takes an analog voltage (ex: 1.2V, 2.7V, 500 mV, etc) and transforms it into digital data that can be transmitted to an MCU to read. This allows the MCU to sense voltage, current and even temperature (the specific implementation will be explained later). Another advantage of external ADCs is that analog voltages are very sensitive to noise, which means that if they are transmitted over a long distance, it will lose a lot of resolution and might be completely different from the intended sensing. On the other hand, digital data is much less affected by transmission distance and thus, the values will remain true to the intended reading.

This section will look at one particular IC that was used in ODIN: the ADC128S102CIMTX. It is a 12 bit ADC, which means it samples the voltage range from 0 to VA and divides it in 2^12 (8192) measurements. For a VA = 3.3V, this means that it can sense the analog voltage up to 0.8 mV precision. Without going into detail, it can be understood that a 0V reading will be translated to 0000 000 000, a 0.0008V reading will be translated to 0000 0000 0001 and so on. These strings of data are sent by serial communication to the MCU. The particular protocol used for this is defined by the chip, which means that during part selection, the chip can be chosen with its serial communication type in mind. In the case of the ADC128S102CIMTX, SPI is used and it will be required to implement proper device and bus protection (reference [1]).

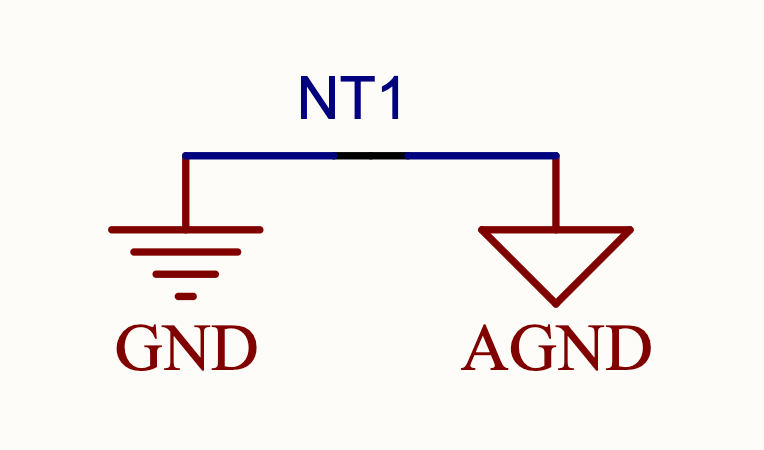
Powering it

The first step in implementing the ADC chip is to properly power it. This should always be the first step in any design. It is especially important for ADCs in general because as mentioned earlier, analog voltages are very sensitive to noise and improperly powering the device will introduce a lot of noise. The way to ensure low interference is by splitting the analog and digital power. This can be seen on the chip’s schematic diagram.



**Figure 1: ADC128S102CIMTX schematic diagram**

Two separate power rails are present: DVDD + DGND and AVDD + AGND. The A and D stand for analog and digital, respectively, and this allows the design to be separated. The digital power and ground can be connected to the main power and ground of the board and DVDD is required to run at 3.3V. The analog power, on the other hand, can be connected to the main power through much more filtering, usually using a ferrite bead and decoupling capacitors. The analog ground is the one that requires a bit more consideration: it can be directly connected to the main ground, but only at one singular point. The way to implement this on a schematic is using a net tie seen in Figure 2. It is also good practice to use net labels to separate VA and VD since you want to use VA as the analog reference.



**Figure 2: Net Tie to Connect GND & AGND**

This allows the two grounds to be connected directly together, but only in one point. (most often through a 10 mil track). The advantage of having a single connection point is that noise on the ground plane will not reach the analog ground as much. The PCB layout will be looked at in further detail at the end of this section. Lastly, both the analog and digital input require decoupling capacitors. The datasheet here recommends a 1uF and a 100nF on each input pin, but this might differ depending on the specific chip.

Channel connection

The second step is to connect the serial communication ports. To do this step, refer to reference [1]. It explains the implementation of SPI and its protection circuit. This specific chip has 8 channels for analog input, but this can vary depending on the chosen chip.

The next step is to properly implement the ADC inputs, i.e. IN0 through IN7 in Figure 1. The first thing to keep in mind is that the analog inputs cannot exceed the VA value. This means that if the VA input is at 3.3V, any higher voltage fed to the analog input will trigger internal protection and output wrong values. The second consideration regarding the analog input is that they need to be kept as clean and noise free as possible. To do so, the easiest way to filter most of the noise is to implement a low-pass filter on every analog input. The datasheet for the chosen ADC recommends a cutoff frequency of 40 KHz, but since we want to filter out as much noise as possible, and also use values commonly used across our designs, it is recommended to use a 1K ohm resistor with a 100nF capacitor. This gives us a cutoff frequency of 1.6 KHz, which should very effectively attenuate all AC noise.

This last section will look at three different ways to use the analog inputs to read voltage, current and even temperature. As mentioned before, all 3.3V references should be replaced by the net label assigned to VADD

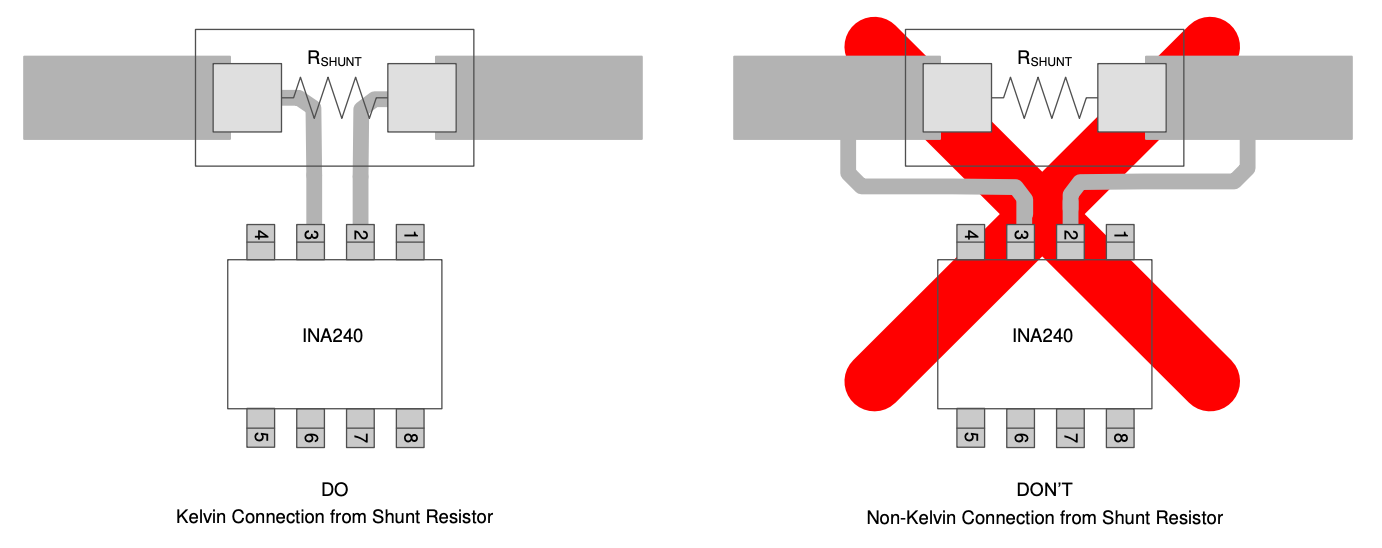
Voltage sensing

This is the easiest one to implement, since all that is required is to feed the power input more or less directly into the analog inputs of the ADC. There are a few things to keep in mind: the input should never exceed VA as previously mentioned, so attempting to sense a 5V line with a VA of 3.3V could damage the inputs if not implemented properly. The simple solution to this is to create a voltage divider using high resistance to split the voltage. Two 100K ohm resistors is pretty standard for such a voltage divider. Lastly, it is important that the previously mentioned filter is implemented to protect the input, as the 1K resistor in line will not only enable the low pass filter to work, but also limit the current being fed from the power line.

Current sensing

Current sensing is the more tricky one to implement. The ideal current meter used in theoretical classes does not exist in reality, so another way to measure the current must be used. One option that comes to mind would be to put a resistance in the power line and measure the voltage drop across to calculate the current. Although this would work, it would also affect the power transfer of the line since getting a good voltage range implies a high resistance. To solve this, a low resistance should be used, typically in the range of 50 to 100 mOhm. These resistors are commonly called shunt resistors. With that said, another issue arises from this: the voltage drop across such a small resistance is very small, in the millivolts range or even less. Feeding this directly to the ADC inputs will not yield a high resolution which is less than optimal. This is why there is one more element to add to this: a current sense op-amp.

Current sense op-amps are operational amplifiers designed to serve as current sensors. The principle behind them is quite simple: it takes the voltage difference between its two inputs and amplifies it. This is perfect for current sensing since all is required is to connect the inputs across the low resistances and the small voltage drop generated will be greatly amplified, resulting in a much better resolution on the analog read. It is important to keep a Kelvin connection methodology around the small resistance, also called shunt resistance, which optimizes the analog read. Kelvin connection to a shunt can be seen in figure X.



**Figure X: Kelvin Connection to Shunt Resistor**

## 

**Figure 3: MAX4373 Current Sense Implementation**

There are a few things to note on Figure 3. First, two 100 mOhm resistors were used to equate a 50 mOhm resistance. The reason for this is because this particular 3.3 power line has a typical current consumption of 1 to 1.2 amps. All this current shoved into a single resistor would result in a lot of heat generated, which could end up reaching above the rated temperature and damage the component. Instead, two resistors are used to split the power dissipation in two. Another thing to notice is that this is a unidirectional current sense amplifier, which means that RS+ must always be higher than RS-. Bidirectional op-amps are available but are usually more costly and unnecessary for most designs. Lastly, the gain of the op-amp is the most important parameter to consider when doing part selection. It needs to be chosen to provide an optimal voltage range on the output, increasing the resolution of the ADC. It is also good to note that since the output current of the op-amp is negligible, it is not necessary to take into account the voltage drop across the filtering resistor.

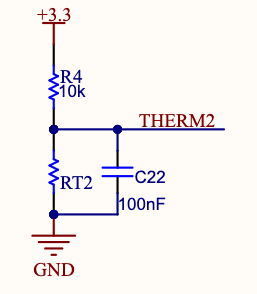
The math behind this is not particularly complicated, but it still needs to be understood. First, the voltage and maximum current consumption of the line needs to be known. In the case of Figure 3, the maximum consumption is 1 to 1.2 Amps. This is calculated by adding the maximum power consumption of each device on the line. Then, the voltage drop across the shunt resistor needs to be calculated. Since V = IR, it can easily be determined that the maximum voltage drop across the shunt resistor is 60 mV. Now, the goal will be to amplify this voltage to utilize as much of the available voltage range of the ADC as possible. Since the ADCs in our design use 3.3V as the VA, the op-amp needs to have a gain that would get it close to this. The MAX4373 has different variants, with the one in Figure 3 having a 50 V/V gain. This is perfect since 60 mV \* 50 V/V yields 3V on the output. Although it does not quite reach the full 3.3 range, it is more than acceptable given the high resolution obtained from this.

The last step to this lengthy current sensing implementation is to inform the software team of the math behind the design. In the case of Figure 3, with the op-amp gain at 50, and the shunt resistance at 50 mOhm, the following equation can be derived to obtain the current from the read voltage:

From this equation, any current can be determined by reading the analog voltage on the chosen input. The values of the gain and the shunt resistance should be adjusted to obtain the best resolution possible, which means the output voltage of the op-amp should be as close as possible to VA.

Temperature sensing

Temperature sensing is also pretty straightforward to implement. It only requires one particular component: a thermistor. Without going deep into the physics of this component, it is simply a resistor that has a variable resistance based on temperature. They have a base value at 25°C, usually 10K, and a temperature coefficient that determines the change in resistance with the change in temperature. The only challenge with this component is to figure out how to read this change and turn it into temperature sensing. It turns out to also be very straightforward: form a voltage divider using a thermistor and a regular 10K, and read the divided voltage. When the thermistor’s resistance changes, the ratio between it and the 10K resistor also does, resulting in a change of voltage on the divider.

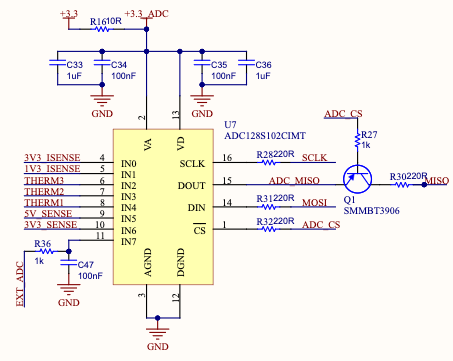


**Figure 4: Typical Implementation of a Thermistor**

RT is the designator used for thermistors. THERM2 simply reads from the divider and the temperature can be determined from that reading. The math behind extracting the temperature from the voltage reading will have to be determined using the temperature coefficient of the chosen thermistor, which can be found in the data sheet. Note: The 100 nF capacitor is only there for filtering purposes.

Here is an example of the math behind determining temperature based on the voltage input from THERM2. Va is the reference voltage and RTcoefficient can be determined from the data sheet of the specific thermistor used.

Figure 5 shows the final implementation of the ADC. Note that ISENSE stands for current sense, and THERM stands for thermal sensing.

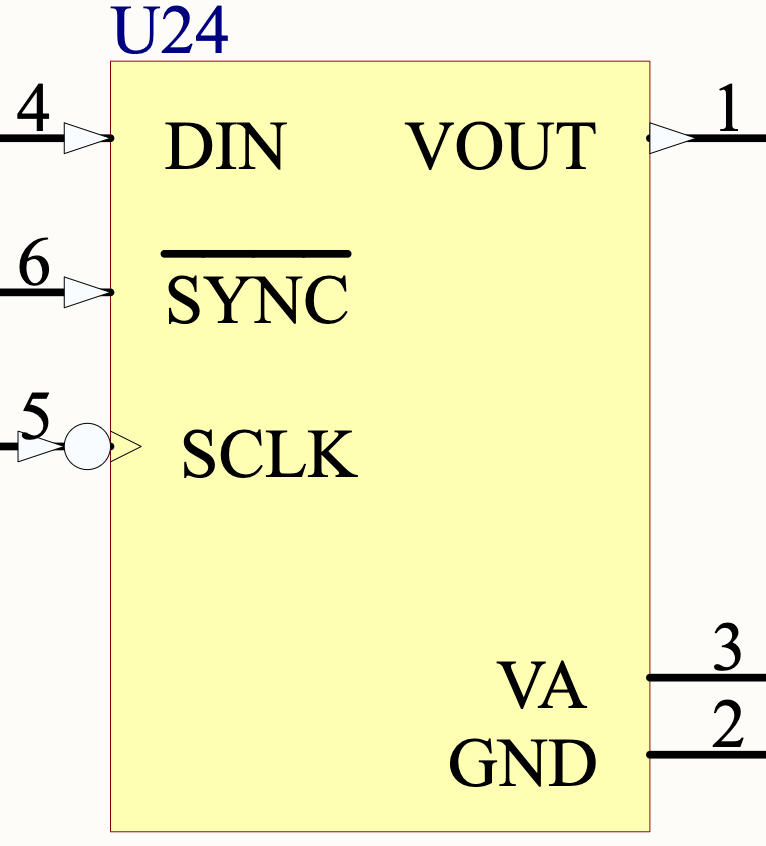


**Figure 5: ADC Implementation**

## DAC

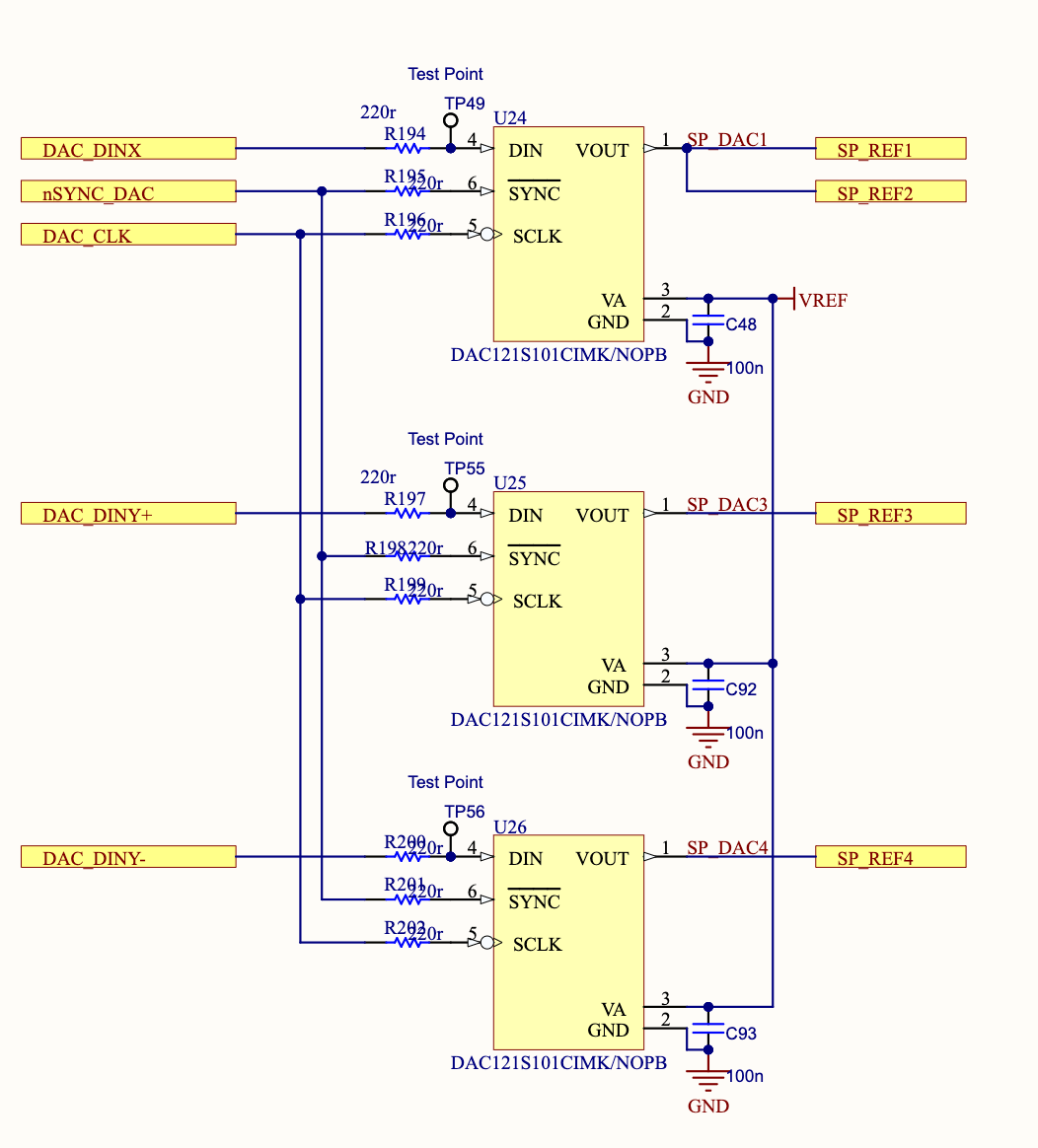
Digital to analog converter, or DAC, is the exact opposite of an ADC. It takes a digital signal as input and outputs an analog voltage. This can be used to control specific circuitry that depends on an analog voltage to modulate its functions. The digital data can be transferred via virtually any protocol as long as the selected chip supports it, and the DAC will convert it to an analog signal with a precision based on the chip itself. Much like the ADC, it will have a precision rating, for example 10 bit, which means it can adjust the analog output up to Vref/2^10 precision. They also have a certain amount of channels available depending on the specific chosen IC.

This document will use a simple example of a DAC implemented on the power supply system of SC-ODIN. DACs are very specific to certain implementations, so this is the only time it was used for previous projects. With that said, it is still good to understand how it works and how it can be used for complex designs. The specific chip used on the PSS is the DAC121S101CIMK/NOPB. It is a 12-bit, automotive grade, SPI compatible digital to analog converter. More precisely, it uses a modified version of SPI since DACs do not need to transmit any data on the MISO bus. They simply need to receive the digital signal that needs to be converted as well as the clock signal. The schematic diagram of the DAC121S can be seen in figure 6.



**Figure 6: Schematic Diagram of the DAC121S101CIMK/NOPB**

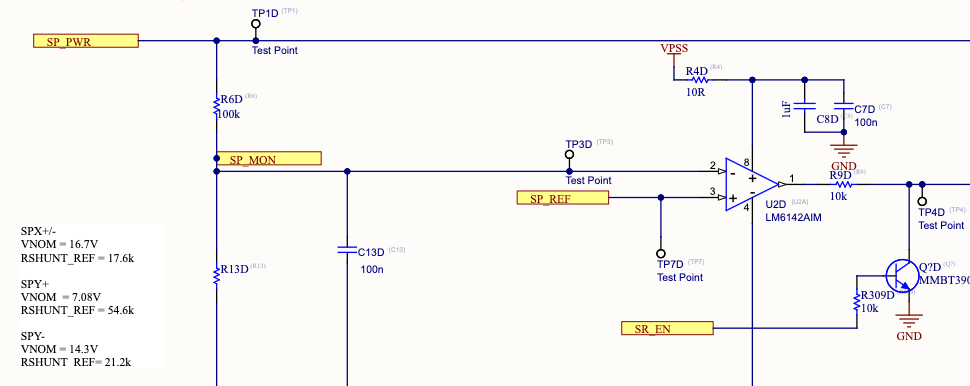
Firstly, the power ports of the devices are VA and GND. VA serves as both the power supply and reference voltage for the DAC, and should ideally be connected to a stable voltage reference that is able to supply current. DIN is the usual slave data input of the SPI bus, which means it needs to be connected to MOSI. SCLK can simply be connected to the SPI clock and VOUT is the output of the DAC, meaning it will be an analog voltage and needs to be kept as clean as possible. Finally, the SYNC pin is associated with the previously mentioned alternative SPI implementation for DACs. It serves mainly as the usual Chip Select of the SPI protocol, but it also has an interrupt functionality depending on the timing of the signal. This should simply be connected to an MCU pin and the software team will be responsible for specific interrupt implementation. Figure 7 shows the overall implementation of the DAC.



**Figure 7: Implementation of the DAC121S101CIMK/NOPB**

In this implementation, all SYNC pins are connected together to reduce the number of chip select pins required and it does not bring any concern for collisions on the MISO bus given they do not output anything. All the references are individually controlled using different data input lines, which is also another feature of this particular SPI implementation.

The implementation of this DAC on the PSS was done to control the shunt regulators. Shunt regulators are a set of circuitry that is able to dissipate a large amount of current if the system input is too high for its power consumption. They are particularly important when dealing with solar panels since they supply power regardless of the state of the system. Without going in depth on the circuitry, the role of the DAC is to adjust a voltage reference to an op-amp, which will in turn adjust the amount of current that is shunted out from the power input. Figure 8 shows a broad overview of the shunt regulator implementation, with SP\_REF behind the reference voltage fully controlled by the DACs.

****

**Figure 8: Shunt Regulator Circuit (first half only)**

The basic principle of operation of this circuit is that a voltage monitor from the solar panels is fed into the negative terminal of an op-amp and the positive terminal is connected to the DAC voltage reference. The solar panel voltage monitor is constantly read by the MCU and if the power input becomes excessive compared to the need of the system, the MCU will control the DAC reference, which will generate a precise output voltage from the op-amp and this will be fed to the gate of a power MOSFET. Depending on the applied bias from the op-amp, the power MOSFET will dissipate a precise amount of power from the solar panel input.

## RTC

A real time clock does exactly what you would expect it to do: it keeps track of the time. Although it sounds like it would be an irrelevant component since time is available on any electronic devices now, it is actually necessary on satellites that will not be in contact with earth at all time. Time tracking is not as simple as one might think, and this is why RTCs are really important for any electrical systems like the ones we produce. It is particularly relevant for MCU designs, as is discussed in the Learning Package on MCU design (reference [2]).

The way an RTC works is quite interesting. The principle behind it is the exact same as a quartz watch that people wear everyday: it utilizes a quartz crystal. Crystals, also known as oscillators, are actual physical crystals that possess particular properties crucial to accurate time keeping. Basically, when a very small electrical signal is sent through a crystal, it will start to oscillate at a very precise frequency. For time counting, the frequency of the crystal needs to be precisely 32.768 KHz. The reason for this is quite interesting: it is the exact frequency that results in a 1 Hz signal when 15 flip-flops are connected in an array. Without going into depth, feeding a clock signal through a flip-flop will make it output a clock signal with half the frequency of the input. Chain this reaction 15 times with a 32.768 KHz input clock will result in a 1 Hz signal at the output. In other words, 2^15 = 32 768. This is how time is measured, since counting a 1 Hz signal is very simple and crystals are very precise so it can be relied upon to keep accurate track of time.

This section will look at a particular RTC: the PCA85063A. It is a pretty straightforward component to implement and includes all the features required from an RTC. Figure 9 shows the schematic diagram of this IC.

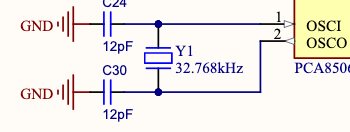
## 

**Figure 9: Schematic Diagram of the PCA85063A**

The diagram for most RTCs will have similar pin layouts: VDD and VSS for power, OSCI and OSCO for the crystal connections and a few connections for data depending on the protocol used by the chip. This particular chip uses I2C, and has an interrupt pin as well as a CLKOUT pin. This is because it has two extra features: a clock divider and an alarm. This will be explained at the end of this section, for now the main features will be looked at.

First, the RTC needs to be properly powered. Fortunately, it does not require any special filtering on the power inputs or ground node considerations. Simply add the usual 100 nF capacitor between the VDD pin and GND. The one thing that is unique to an RTC is that it needs to be powered on all the time, otherwise it will lose track of the time, just like old alarm clocks. To counter this, a simple coin cell battery can be small. RTCs usually consume very little power, so a simple coin cell will last for a very long time. With this particular IC, a CR2032 will last multiple years so there is no need for a big battery. Coin cell batteries also often supply the correct voltage level, in this case 3V, which is really easy to implement.

Second, it is important to understand how to properly connect the crystal. Any mistakes in implementation could result in massive errors and incorrect time keeping. For most RTCs, including the PCA85063, the required crystal is a quartz 32.768 KHz crystal. They come in different formats and precision, but in general most crystals with these characteristics will do. The chosen one for this design is the Q0.032768-JTX310-12.5-10-T1-HMR-LF (crystals usually have very weird names). Figure 10 shows the proper way to connect the crystal.

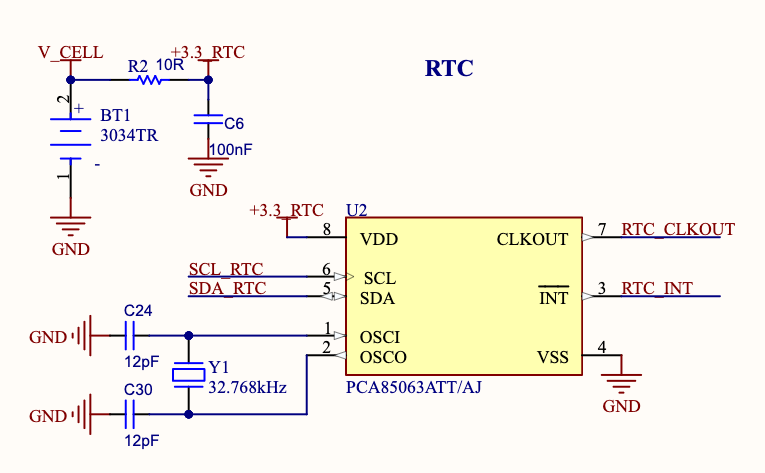


**Figure 10: Crystal Connection to the RTC**

Few things to note on this figure: crystals are usually bidirectional, so there is no specific way to connect the in and out ports of the RTC. The second thing is the presence of capacitors. Unlike one would think at first, these are not decoupling capacitors. The physics behind them is complicated, so all that is really required to know is that they need to be on both sides of the crystal and have a value of 12 pF for 32.768 KHz crystals. PCB layout rules are more complicated when it comes to crystals, but this subject will be talked about in depth in another Learning Package.

Third thing to implement is the data lines. This particular chip uses I2C and will require the usual protection circuit for this protocol. Refer to reference [1] for further details on I2C and its protector. The used protocol will differ depending on the selected IC.

Lastly, this chip has two extra features: the clock divider and the alarm clock. The clock divider is a feature that allows you to output a clock signal that is either exactly 32.768 KHz in frequency, or any fraction of that. This can be useful if other parts of the design require a specific clock signal. Most of the time, it is simply fed to the MCU for it to use. The alarm clock feature is implemented using an interrupt pin that can simply be connected to the main MCU. It is also good to know that the interrupt can be used in pairs with the clock signal for specific applications.



**Figure 11: Implementation of the PCA85063**

A concern that arises from any crystal based component like RTCs is clock drift. Although crystals are generally very precise, external environmental factors can affect their accuracy and introduce small errors. These errors can not only lead to timekeeping issues, but can also cause imprecisions in the system. A great example of this is the ADCS model implemented on ODIN. The model controls the orientation of the satellite during orbit and hence, controls the pointing accuracy of the antenna to the ground station. The estimated angle error from SC-ODIN’s model is 2.22 degrees, with 0.95 of that due to clock drift alone. This shows that clock drift is very much a concern and needs to be addressed. In space, the environmental factors that increase clock drift are all the more present, and corrections need to be put in place. This next section will look at different methods to reduce clock drift error.

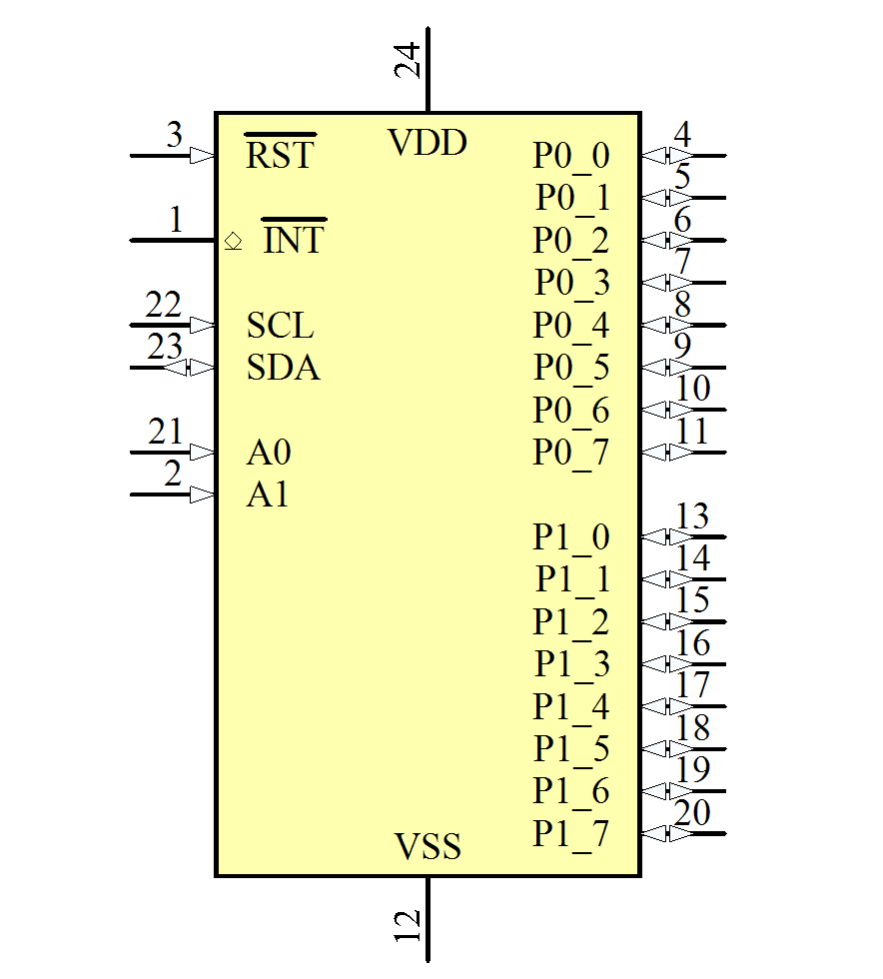
The first method is to compare multiple available clock signals and calculate the drift based on the other clock signals. Once the drift becomes significant, it is possible to reset a clock signal and synchronize it to the rest of the system. This method does not require any outside interaction to correct itself, but it will also eventually have the entire system out of phase with “earth’s time”. The second method addresses this issue, but it requires communication from the ground station to the satellite. It is possible to send a synchronization signal which would adjust all the clocks onboard the satellite to the earth’s clock, thus removing all drift introduced between two communication windows.

## I/O Expanders

I/O expanders are very useful peripheral devices when designing big systems that need to be controlled by an MCU. One of the main issues that occurs with dense and complex designs is that a lot of I/Os are required to control every part of the design, and MCUs have limited ports that can be used. This is especially true when dealing with smaller size MCUs since they have a small amount of pins, with a big majority reserved for power, crystals and communication protocols. The I/O expander, as the name suggests, solves this problem by providing multiple I/O ports that are controlled through a communication protocol. Depending on the specific device, some offer two or three state logic, allowing the I/O expander to not only output control signals but also read statuses and flags.

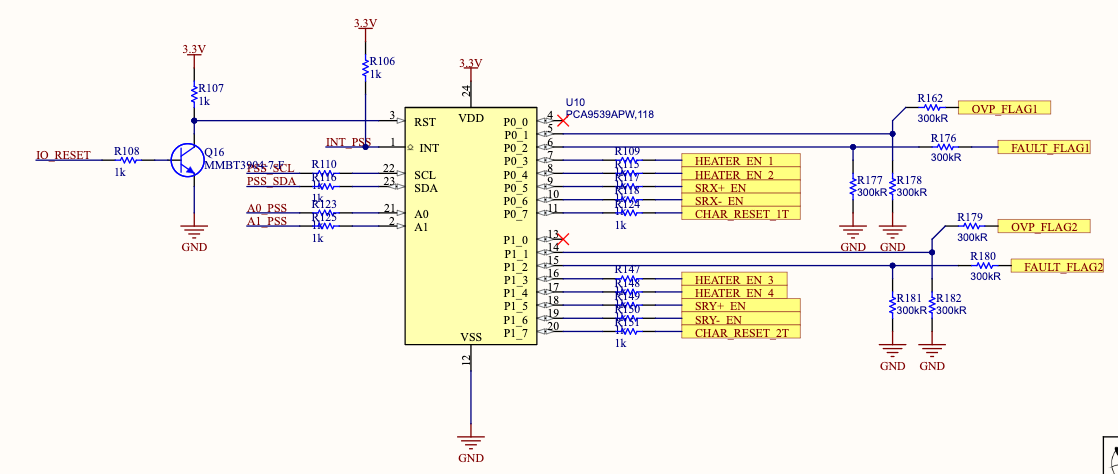
I/O expanders were used a lot during SC-ODIN, more specifically on both the power board (PSS and PDS). Since they are both mainly analog or TTL based designs, they require a lot of monitoring and regulation that digital designs usually integrate using communication protocols. This does not work for analog designs since they require specific control signals to be applied to different sections of the design. This is why I/O expanders are so useful: they allow a digital bus to control multiple aspects of a design and read the provided statuses. This section will look at the I/O expander implemented on the PSS, the PCA9539APW,118, to concretely see how they can be used to control complex analog circuits.

The PCA9539APW,118 uses I2C as its communication protocol, three state logic on all 16 pins and interrupt/reset ports. Figure 12 shows the schematic footprint of the component. The interrupt and reset pin can be seen as well as the I2C ports and the address controls.



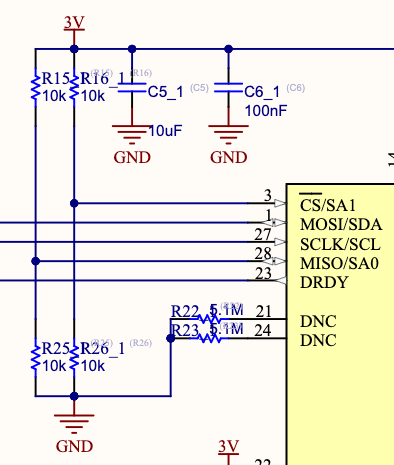
**Figure 12: Schematic Footprint of the PCA9539APW,118**

On SC-ODIN, the PSS is almost fully analog given the reliability of analog circuits for space applications. The issue with analog circuits is that they are bulky, power hungry and require a lot of control signals. In this particular implementation, the I/O expander controlled all heater enables, all shunt regulator enables and the charger resets. It was also responsible for reading the fault signals from the undervoltage, overvoltage and short circuit protections. The full circuit implementation can be seen in figure 13.



**Figure 13: Implementation of the PCA9539APW,118**

There are a few things to note on this implementation. The I2C bus implementation is very standard, with the added feature that the A0 and A1 pins can be used to change the slave address of the device on the bus. Usually, an address is chosen by CDH for each slave device and all that is required is to tie A0 and A1 to ground or VDD using 10K resistors to set the address. For this implementation, they were connected to I/Os of the main computer so the final decision on the address could be done after implementation. It is important to note that it is not good practice, and all the addresses should be planned out in advance. The proper way to set up I2C address selectors can actually be seen on figure x, which is how the magnetometer address selection was set. In comparison with the implementation of the I/O expander, it simply requires two extra resistors that will be left DNP instead of using 2 full I/Os on a microcontroller. This is also much more robust as a bug with an MCU can cause the device to change address and create issues on the I2C bus.



**Figure 14: Correct Implementation of the I2C Address Selector**

R15, R16 are used to tie the address pins to 3.3V and R25, R26 are used to tie the address pins to GND. To select the address, simply leave 2 of the resistors DNP to write the address in binary.

Another important thing to note on figure x is the pull up resistors present on both the RST and INT given they are both active low. They both need to be connected to I/Os of the controlling MCU of the design to allow CDH to control the device appropriately. Finally, it is important to put proper I/O protection on each pin to ensure that the control signals don’t damage the component. In general, a simple 1K resistor is required for all the I/Os that will be implemented as outputs (like P0\_3 to P0\_7 and P1\_3 to P1\_7 on figure x) and a higher resistance, usually above 100K for I/Os that will be used as inputs (P0\_1, P0\_2, P1\_1 & P1\_2 on figure x). It is also important to add pullup or pulldown resistors on input channels depending on the polarity of the status signal.

## 

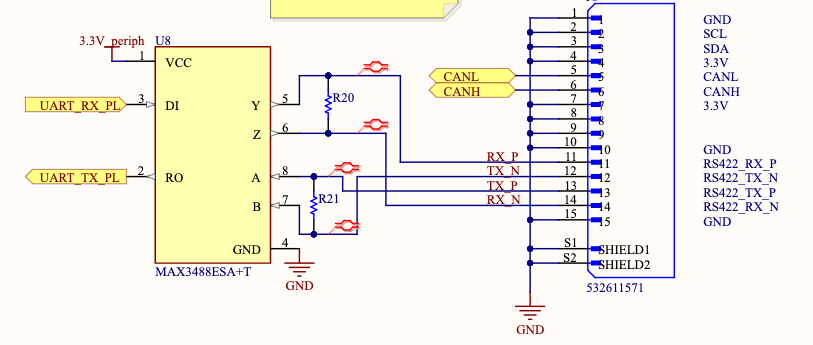
## 

## Transceivers

Before beginning this section, it is highly recommended to know at least the basics of serial communication (reference [1] for the basics). Transceivers are devices that use a certain type of data and transmit it in a different way depending on the requirement of the design. For example, some transceivers can take a certain protocol as input and transmit using another protocol. This is particularly relevant considering different protocols can transmit over different lengths, and a transceiver allows a design to transmit over the limitation for length of a particular protocol. Some are also able to transform common mode signals into differential mode signals, which are usually more resilient to noise and can transmit over greater distances.

One of the most commonly used transceivers is a CAN bus transceiver. The reason for this is quite simple: a lot of MCUs will not have an integrated transceiver, but rather a CAN\_RX and CAN\_TX line. To solve this problem, a CAN transceiver is used. This can be seen in the CAN Bus section of reference [1].

This illustrates the main function of transceivers: take a certain type of data as input, often from protocols like UART and RS232, and transform them into differential signals like CAN, LVDS, RS485 and more. Another example of this is the MAX3488. This chip, depending on the package used, is able to communicate from RS232 to RS422 with controlled slew rate for EMI reduction, driver protection and increased line length. Figure 15 shows the implementation of this chip to allow an MCU to transmit to the main imager of the satellite. The MCU in question did not feature RS422 communication, so the transceiver was used to convert one of its UART ports (which is very similar to RS232) to the required RS422 protocol.



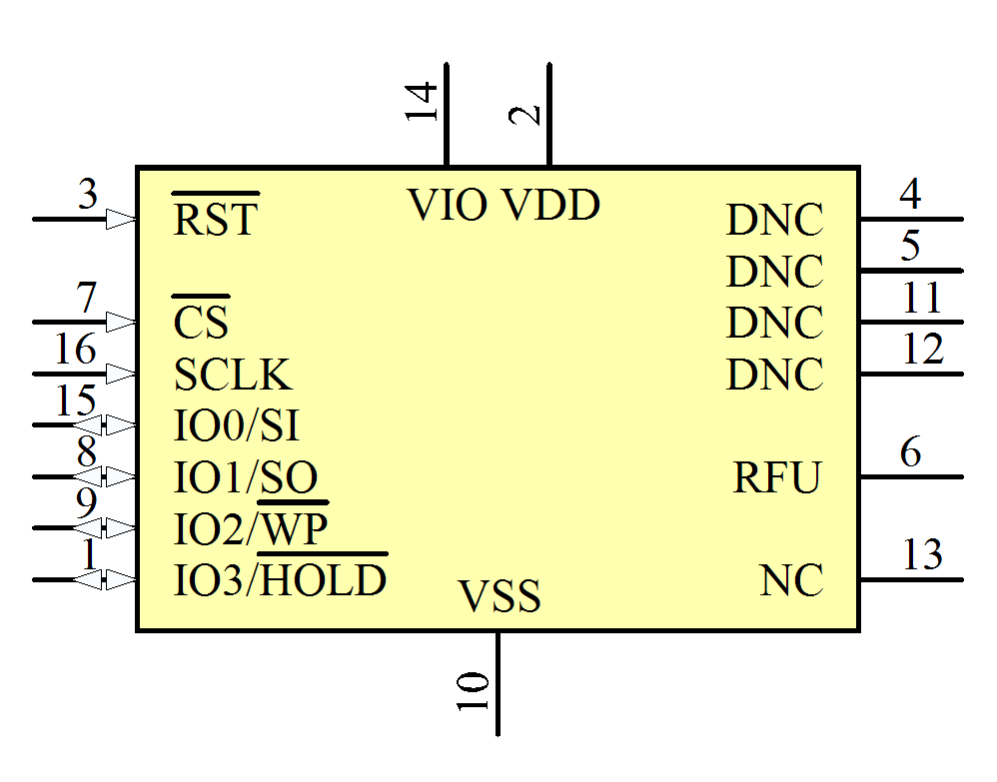
**Figure 15: Implementation of the MAX3488**

The nice thing about transceivers is that they are generally very easy to implement: connect it to power, connect the input protocol and connect the output protocol. The only thing to consider is that some protocols will require bus terminations, hence the presence of R20 and R21. These will either need to be matched to the overall bus impedance, or put to a high resistance if no significant power transfer is required.

## Flash Storage

Flash storage is commonly known as non-volatile memory that has fast data access rates, hence its name. This type of memory is required for any data, programs and models that need to be saved even if the power is turned off. There are two main types of flash storage: NAND flash and NOR flash. Without getting into the details, NOR flash is usually faster to read than NAND, but it's generally more expensive and slower to write or erase data. For this reason, NOR flash is prioritized in our design since there are very few cases when data will need to be written while in operation. It is important to note that MCUs have internal flash memory, but it is generally very small. The main MCU used on SC-ODIN has in the hundreds of Kilobytes of flash storage, which is not enough for the main code and the control algorithms to be loaded on. For this reason, external flash memory chips are added and connected to the MCU.

This section will specifically look at the S25FL512S chip, which was the NOR flash used on SC-ODIN’s flight computers, the SPEAR-M7. This chip has 512Mb of storage, which is significantly more than the internal memory of the main MCU, and features SPI interface and AEC-Q100 rating (automotive grade rating).

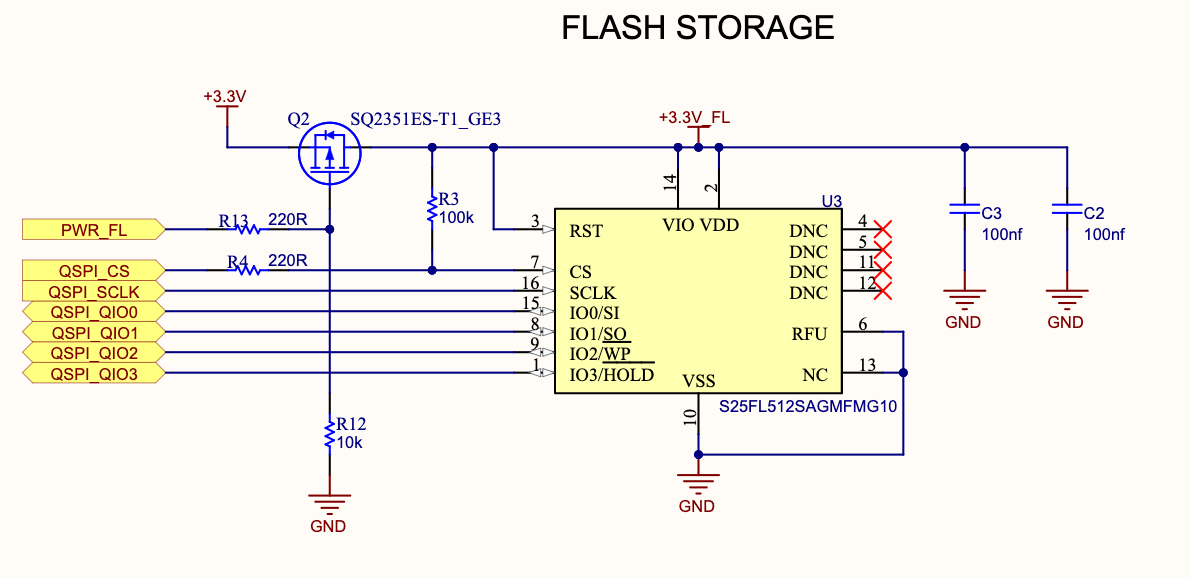


**Figure 16: S25FL512S’s Schematic Diagram**

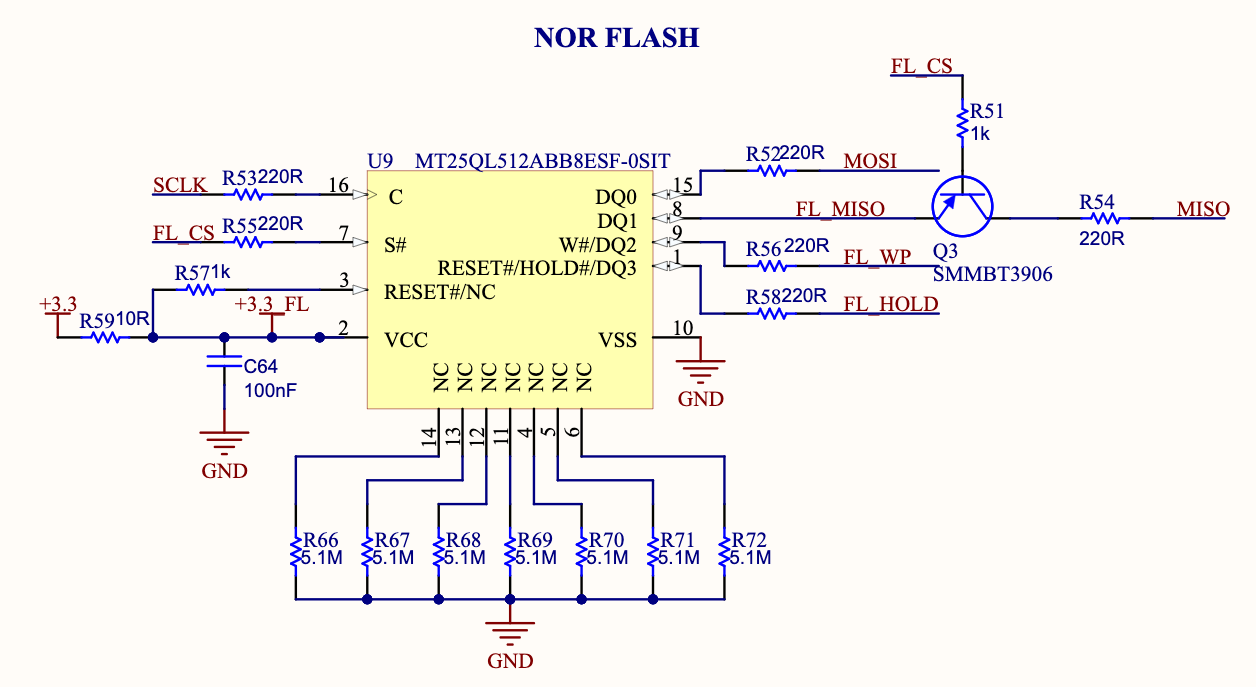
The featured SPI interface of this chip makes it fairly simple to implement. The two power pins, VIO and VDD, can both be powered by the 3.3V node, with VSS as the ground pin. VDD is the core voltage supply, which should always run at 3.3V, and VIO is a versatile power supply that can be used by the IOs to communicate with different voltage level signals. In this case, SPI runs at 3.3V so no need to connect VIO to a different source. CS, SCLK, IO0/SI and IO1/SO are the usual four SPI protocol pins. DNC stands for do not connect, and should be left unconnected for ground equipment, and grounded through a 5.1M ohm resistor for space applications. NC stands for not connected and can simply be grounded. RFU is simply a reserved pin for future uses and is not applicable to our design, so it can simply be grounded. RST, which is active low, is the reset pin and can either be connected directly to VDD or to an MCU I/O with a pullup resistor. This is important since simply connecting it to the MCU I/O does not guarantee a base level high, and will trigger the reset pin over and over until the MCU intervenes. IO2/WP should be connected to an MCU I/O, since it can be used by the software team to adjust the mode of operation. Although this pin is also active low, the datasheet specifies that internal pullups are present so no need to add external ones. Lastly, IO3/HOLD allows the MCU to pause the data transfer, so it should also be connected to one of the MCU I/Os. Once again, this pin is active low and has internal pullup resistors.

A great feature of this particular chip is that unlike classical SPI devices, this chip allows IO2 and IO3 to become additional data transfer lines for increased data speed. This works only because the particular MCU used has pins that have access to every internal peripheral, so any I/O to which IO2 and IO3 are connected can be used for SPI communication.

There is one last feature that is desirable for this sort of device, but it will need to be implemented externally. This feature is the ability to deactivate and power cycle the chip. Intuitively, the reset pin does exactly that for most devices. Unfortunately, in the case of a memory chip, holding the reset line low will also erase the bank address register, which is not something we necessarily want to do everytime we power cycle. To solve this issue, a switching mechanism needs to be implemented before VDD in order to block power input, but it also needs to not trigger the RST pin. The best way to do this is to simply add a mosfet in front of the power input and control its biasing through an MCU channel. Figure 17 shows that implementation. Note that RST is tied after the mosfet, but since it has an internal pullup resistor, it will not trigger a reset. Figure 18 shows a different implementation of a different NOR flash chip, to be used as reference.



**Figure 17: S25FL512S Implementation on the SPEAR-M7**



**Figure 18: MT25QL512ABB Implementation**

## Random Access Memory (RAM)

Random access memory, commonly known as RAM, is the opposite of flash storage: it is volatile memory. It is used to store variables and other types of data that will be reused during a process, which means it has a much higher transfer rate and easier access to memory. With that said, the data cannot be stored after a loss of power, so these chips are only dedicated to active processes. This is also why it is only 4 Mb in size compared to the 512 Mb of the flash storage. MCUs also have internal RAM, but it is once again very limited in size and is not enough for controlling an entire satellite’s operations. This was especially true for SC-ODIN, since all the computing power was on a single MCU, so two MRAM chips were required.

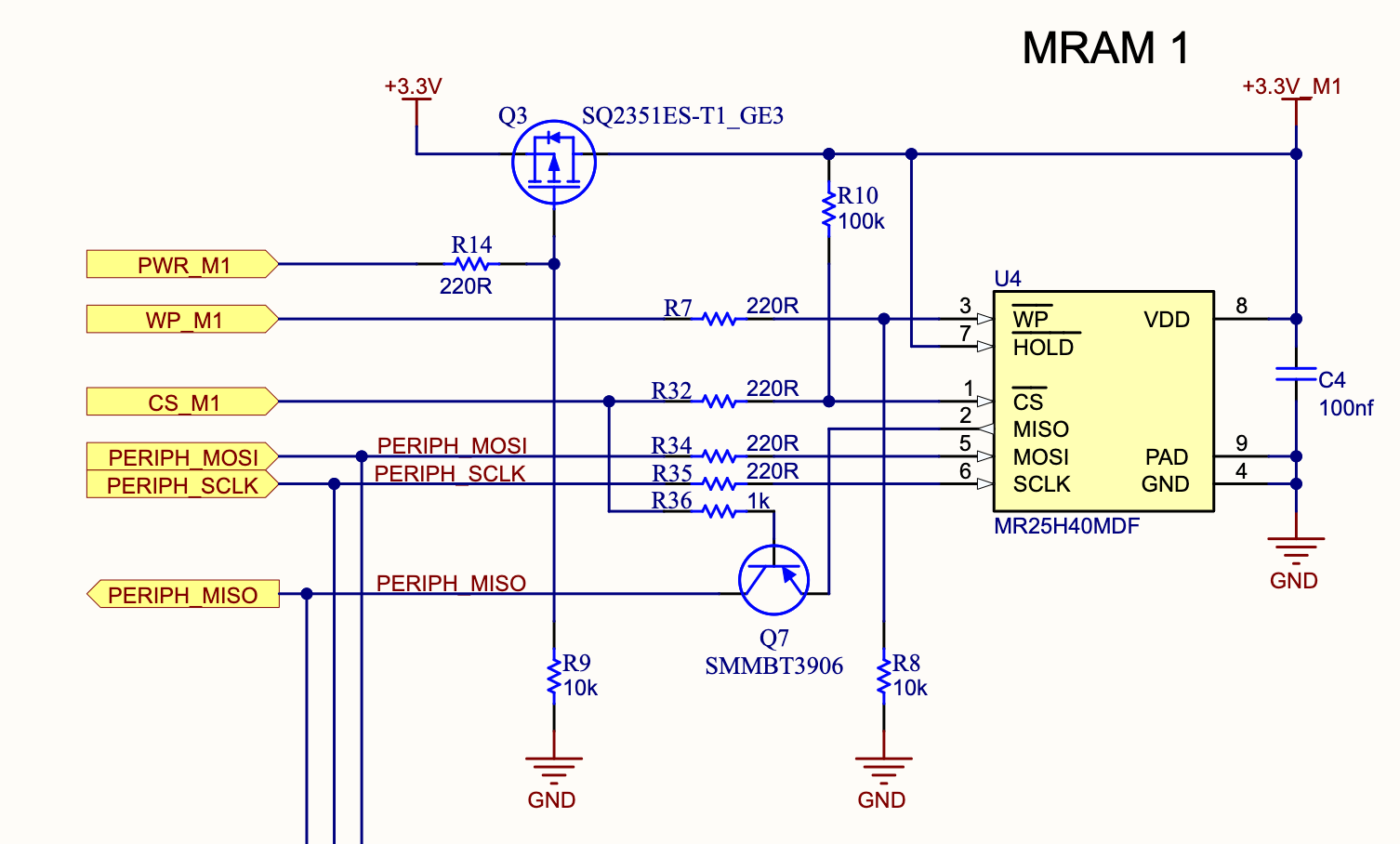
This section will look specifically at the MR25H40 chip, which is an MRAM chip. MRAM stands for Magnetoresistive Random Access Memory, and it is the required type of RAM for satellite applications. This is because in space, radiation is not shielded by the earth and affects the electrical much more. At random, some particles can hit chips and create a “bit flip”, which essentially transforms a random bit from a 1 to a 0 or vice versa. Although this seems insignificant at first, it can be massively disruptive on RAM, which is at the base of any processes running on the MCU and could have fatal effects. This is why memory that can withstand these radiation effects is chosen for space application.

The electrical implementation of these chips is very similar to the implementation of flash storage: it uses a communication protocol, like SPI, has the WP and HOLD pin, and the power pins.

## 

**Figure 19: MR25H40 Schematic Diagram**

The only added pin on this package is the PAD pin. This simply is a thermal pad located under the component to dissipate heat, and should be connected to ground for optimal heat transfer. The rest of the pins work the exact same way as the MT25QL512ABB. Once again, a power cycling feature should be implemented to ensure no damage is done to the memory on reset.



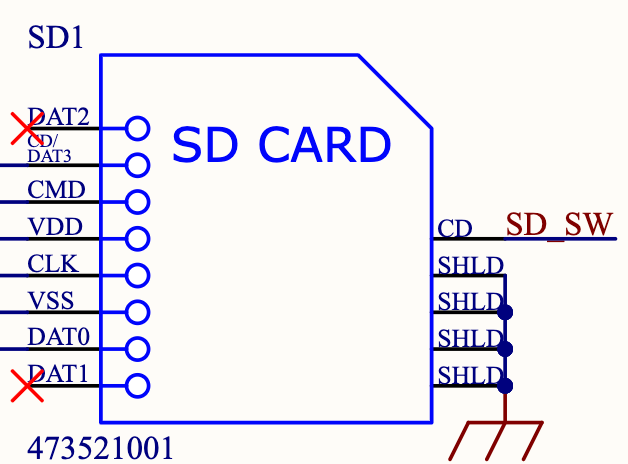
**Figure 20: Full implementation of the MR25H40**

Few things to note on this figure: first, the SPI protection circuit is implemented and can be used as reference with reference [1]. Second, the HOLD pin is tied to VDD to reduce the possibility of a bit flip, which would disrupt the use of the RAM chip and could greatly influence the process in progress. Being tied up to VDD ensures it is always held high unless a power cycle is in progress.

To reiterate, the challenge with the RAM chips is not the schematic implementation, but rather the parts research and eventually, the PCB layout. It is important that the selected chip is magnetoresistive, has the proper automotive rating, good operating temperature, compatible communication protocol and enough memory size for the given purpose. For the last property, the CDH team is the one responsible for defining the requirement of their algorithm and code, so they should be the ones defining this criteria.

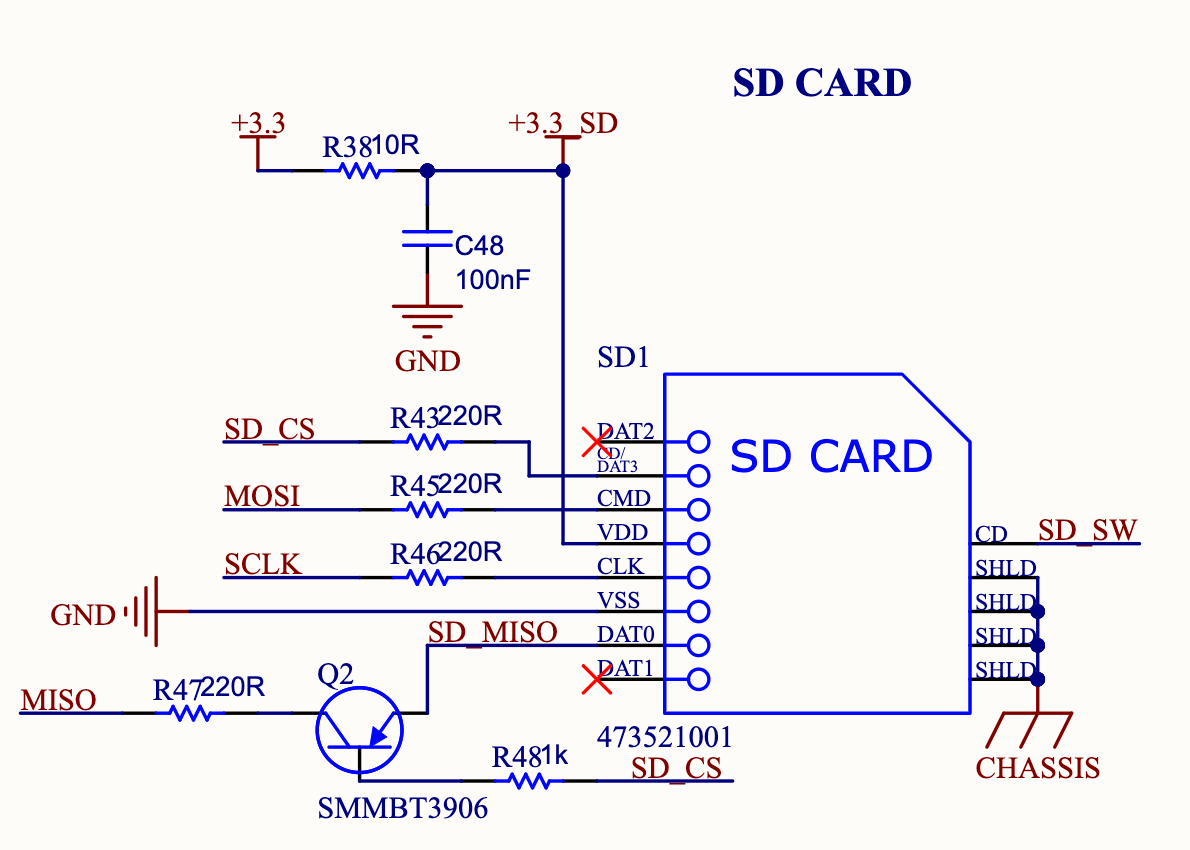
## SD Card Reader

Although this is not a component that will be present in most satellite designs, it is still a relevant component to understand for ground equipment implementation. Fortunately, just like the RAM chip, it is a really easy peripheral device to implement depending on the type and range of functionality chosen. Reference [4] explains in depth the different types and the extra functionalities available.



**Figure 21: Schematic diagram of the Molex 473521001 SD card reader.**

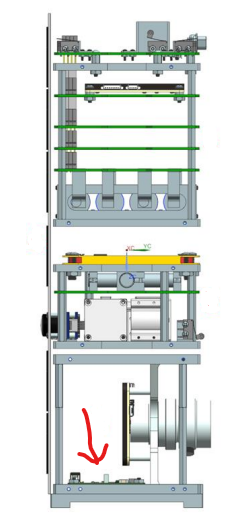
This particular chip uses SPI as its communication protocol. DI or CMD is the MOSI line, DO or DAT0 is the MISO line, CLK is obviously the SCLK line and CD/DAT3 is the CS line. VDD is power and VSS is ground, just like the two previous peripherals. DAT1 and DAT2 are unused pins in this implementation. The only differences with what was previously seen in this document is the presence of a CD pin and four SHLD pins. CD stands for card detect, and it is used to relay to the MCU whether a card is inserted or not. It is a simple mechanism that will ground CD when a card is inserted, which means that a pullup resistor is required on CD to operate properly. SHLD stands for shield, and since an SD card reader is a sort of connector, the mounting connections need to be connected to CHASSIS. This is why the earth symbol is connected to all four pins and named CHASSIS for distinction from GND.



**Figure 22: Implementation of the Molex 473521001 SD card reader**

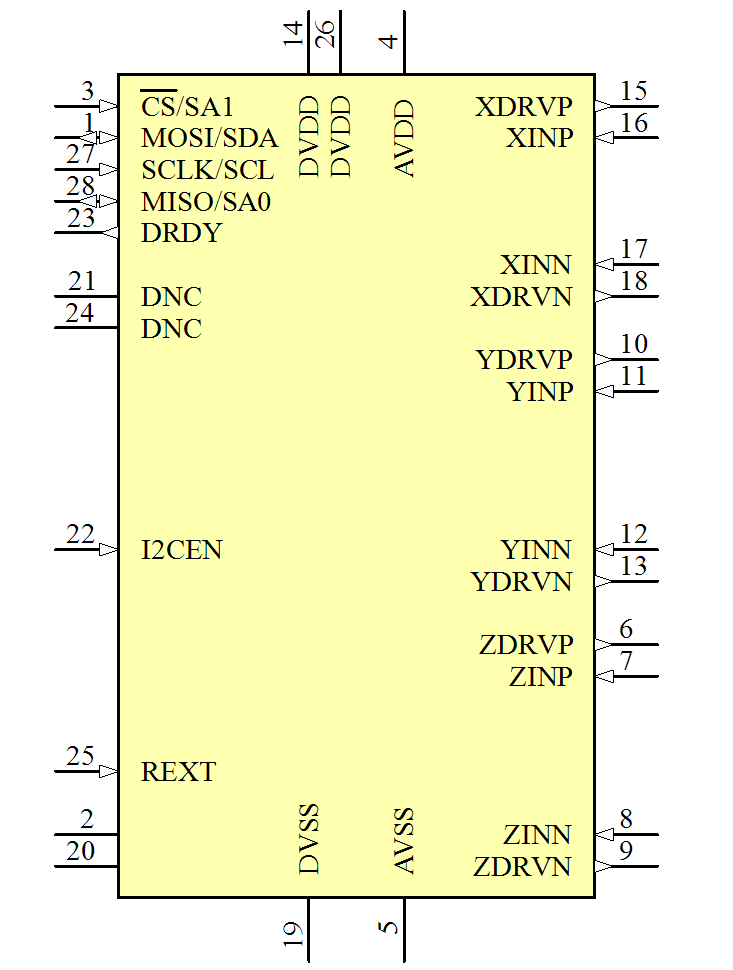
## Magnetometer

Magnetometers are one of the sensors that are used to locate and orient the satellite in space. The principle of operation behind magnetometers is quite simple: it simply measures the earth's magnetic field on three different axes and compares this data to give the orientation of the satellite. This comes with a unique challenge compared to typical sensors: it cannot be placed next to devices that will cause additional magnetic fields to interfere with sensors. Unfortunately, most electronic devices create magnetic fields, which means the PCB that holds the magnetometers needs to be isolated from most of the system, especially from the ECU. For SC-ODIN, this design consideration was integrated by moving the magnetometer PCB as far as possible from the ECU, as can be seen in figure 23. The only piece of electronics that is near enough to it is the payload, which should be turned off for a large portion of the mission. With that said, it was still defined that the obtained results from the magnetometers may be wrong during imaging mode and the control system shall take this into account.



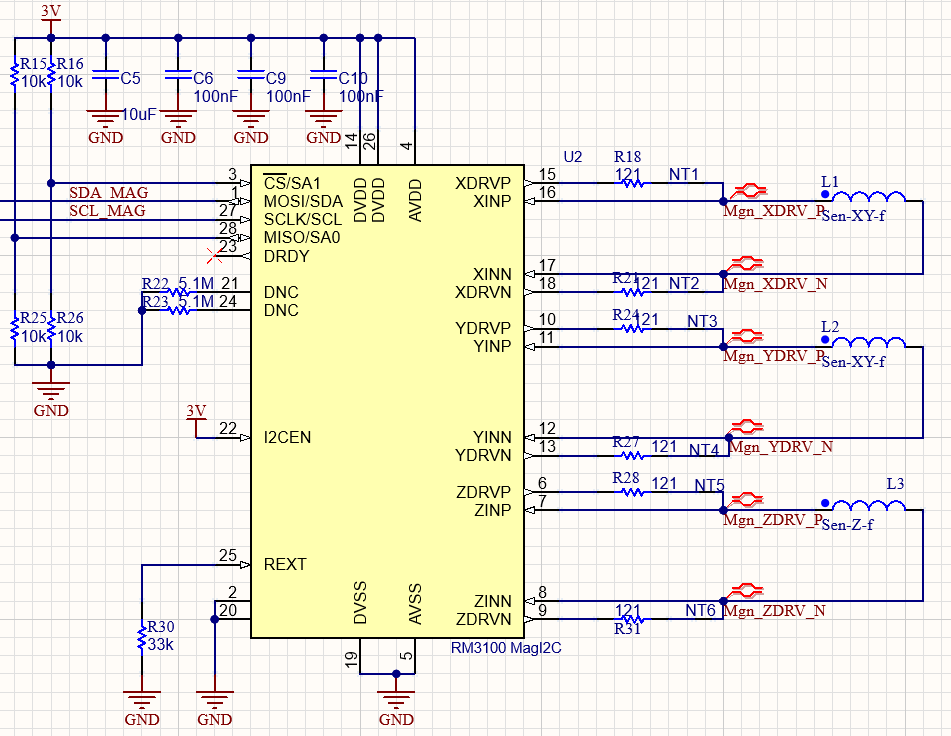
**Figure 23: Position of the Magnetometer Board in SC-ODIN**

The magnetometer PCB is a relatively small and compact board whose only purpose is to hold the magnetometer ICs, coils, provided power and communication lines from the ECU to the ICs. There are a wide variety of magnetometer solutions on the market, but the one that was used in SC-ODIN is the RM3100 suite. The suite includes the magnetometer IC as well as three inductor coils, one for each axis. The IC offers both I2C and SPI communication and either can be used to fit the implementation of the system. Figure 24 shows the schematic diagram of the RM3100 IC.



**Figure 24: RM3100 IC Schematic Diagram**

The IC is pretty straightforward in terms of schematic diagram: it has all the power connections in DVDD, AVDD, DVSS and AVSS, the serial communication ports on the top left and the coil sensors and drivers on the right side of the IC. Figure 25 shows the IC fully implemented.



**Figure 25: Implementation of the RM3100 Suite**

First, the IC needs to be supplied with 3 volts, which means that a converter will need to be added for most electrical architectures to step the voltage down from the typical 3.3 to 3 volts. Fortunately, since it is a low power design and we are only slightly stepping down the voltage, a simple linear regulator will be required for this design. With that said, if a higher voltage is used, a more efficient switching regulator might be better. If uncertain about the difference between both, refer to the learning package on power [5]. The specific converter that was used in SC-ODIN was the TPS76801MPWPREP (datasheet reference [6]) and since the voltage drop is very small, it is required to be an LDO. Once the power conversion is done, all that is required is a bulk capacitance of 10uF and three 100nF capacitors on the power input pins (AVDD and DVDD, one 100nF capacitor per pin). Both AVSS and DVSS can be connected directly to ground.

For serial communication, as mentioned earlier, the RM3100 can use either SPI or I2C depending on the needs of the designer. In the case of SC-ODIN, I2C was used due to the low wire count. It is important to note that to use I2C, the I2CEN port needs to be connected directly to 3V. Otherwise, the default protocol will be SPI. Pin 3, 1, 27 and 28 are used for both the SPI and I2C lines (left of the / is SPI, right of the / is I2C). Since I2C uses two lines less than SPI, the two other lines (A0 and A1) are used to set the address of the device on the I2C bus. This explains the presence of R15, R16, R25 and R26 to set the address (place two resistors total so A0 and A1 can read a 1 or a 0).

Next, the DNC pins should follow the same design methodology used for previous NC and DNC pins, which requires all pins of this type to be grounded through a 5.1M ohm resistor. REXT should be connected to a 33K ohm resistor to ground per the datasheet specification. Pin 2 and 20 can simply be grounded since they are reserved pins.

The last connections to deal with are the coil drivers and sensor pins. These will be the difficult ones to implement on the PCB since they are required to be of very similar length, thus why they are marked as differential pairs.

## 

## Digital Sun Sensor

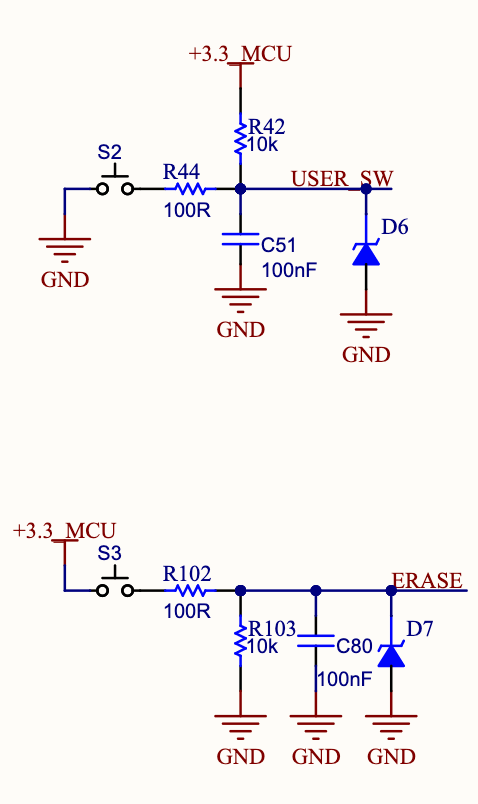
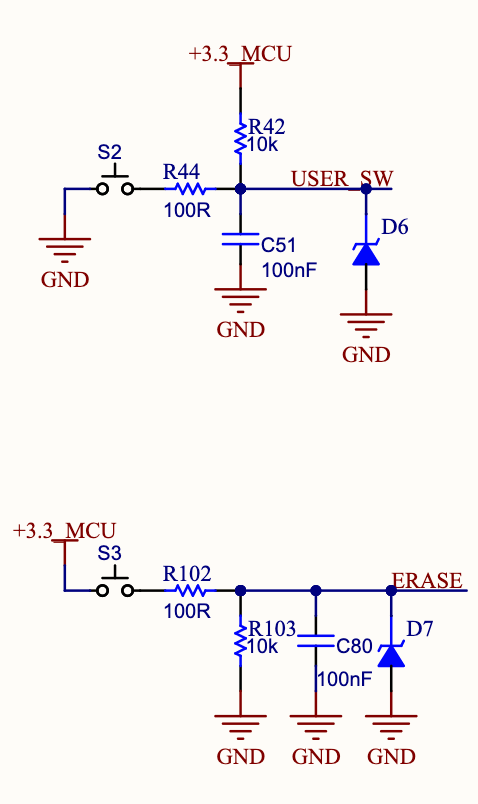
## RADFET

## 

## Push Buttons

Push Buttons aren’t typically defined as “peripherals” but their implementation is still not really intuitive and will be quickly explored in this document. There are a few simple concepts to keep in mind while implementing them, and once understood they can be applied to any push buttons or switches.

This simple component can be used for a variety of applications: trigger logic gates, power cycle, bias transistors, reset or erase an MCU or they can simply be connected to an I/O for testing purposes. In all cases, the electrical implementation is almost the same and will differ in very small areas. Figure x shows the typical implementation of pushbuttons, one is highside and the other is lowside.



**Figure x: Highside and lowside push button implementation**

First thing to note on figure x is the presence, in both cases, or a 100R resistor in line with the button. When the button is pressed, it acts as a direct short on both terminals, connecting either ground or 3.3V to the ERASE or USER\_SW pin directly. This resistor simply limits the current going into the button, preventing it from burning or breaking.

Second is the presence of a 100 nF capacitor. This capacitor is required to prevent bouncing of the button. This problem is due to the mechanical aspect of a button: once pressed, the metal contacts tend to bounce from one another and create a “jitter” effect on the line. This can negatively affect certain functions that might count the amount of times the button is pressed. Since a single press can create multiple bounces, the software will increment the number of times pressed multiple times for a single press, which is not what is wanted. To solve this issue, there are two solutions: either the software team needs to implement a countermeasure to this, which oftentimes simply involves stopping the reading for a short period of time after a press, or it can be implemented using hardware (more details in reference [3]). This is where the capacitor comes in: in the same way a decoupling capacitor works, this 100 nF will slowly drop or increase the voltage of the line, thus preventing sudden jitters caused by bouncing of the button. This is called debouncing, and is always good practice to implement even with proper software countermeasures.

Third aspect is the presence of pullup or pulldown resistors in R42 and R103. Although this is not specific to buttons but rather to the pin it is connected to, it is much easier to ensure that it is not forgotten in the design process. The pulldown resistor is present on the highside button to ensure that the line is held low unless the button is pressed, and vice versa for the lowside button.

Lastly, since push buttons are a component that will often be touched by the user, it needs to have proper ESD protection. This can simply be implemented using TVS diodes, which are zener diodes specifically designed to protect against ESD. The particular one chosen here has a reverse breakdown voltage of 4V, which will never trigger during normal 3.3V operation, but will short any voltage beyond that directly to ground, protecting any circuitry connected to the button.