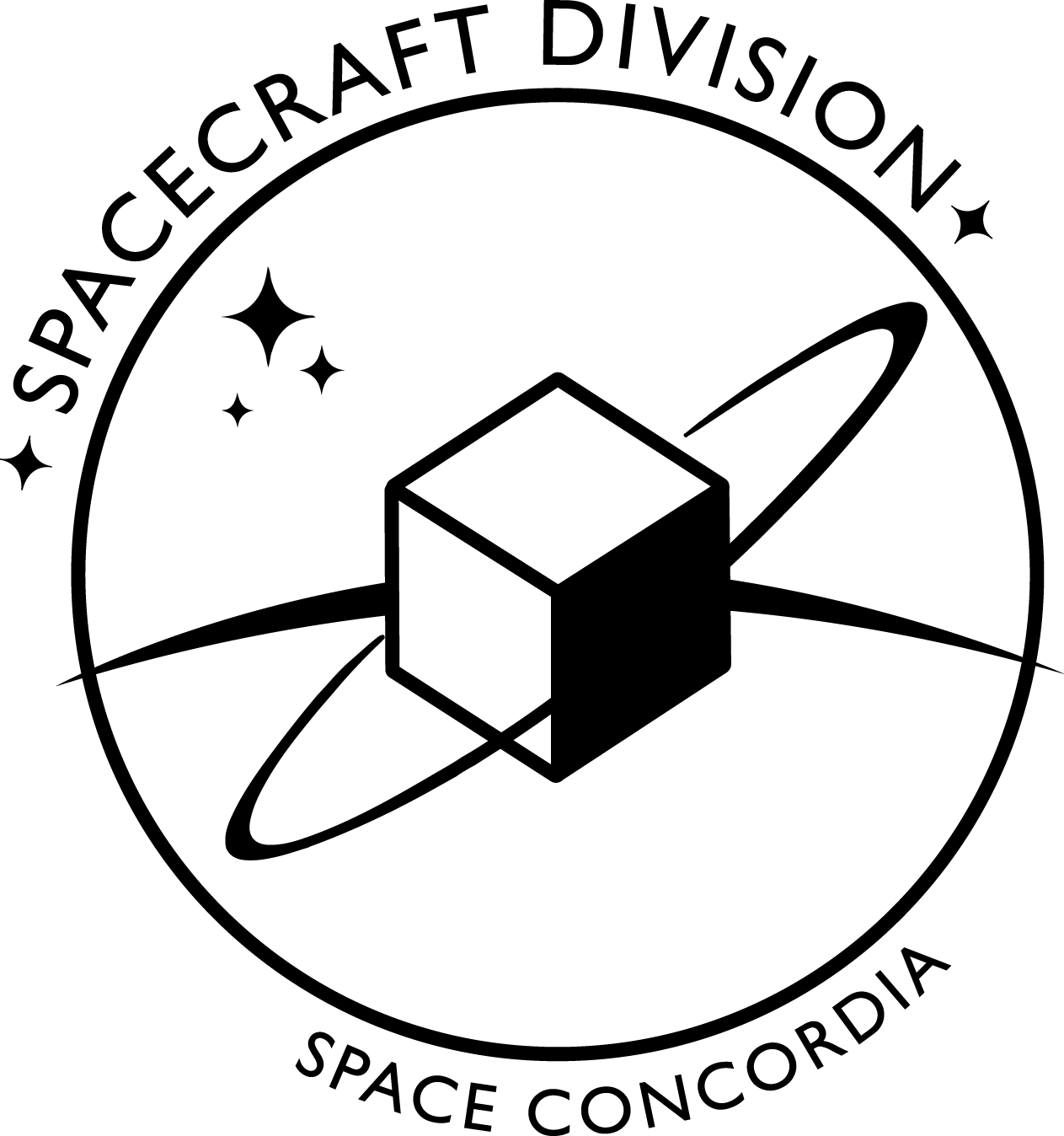
SCSD-LP-EPS-004-A

MCU Design



Prepared by

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

Electrical and Power System Lead

Feb 1, 2023

# **Revision History**

| **Date** | **Revision** | **Changes** |
| --- | --- | --- |
| 5-Feb-2023 | A | Initial Release |

# **Table of Contents**

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

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

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

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

[**Introduction 6**](#_meu3f6nbbwk8)

[**MCU 7**](#_3kwz5opnz7s6)

[Power 7](#_tc10rrylvlur)

[Crystals 13](#_82apm3xaqf7j)

[Internal Peripherals 14](#_nhku6ls1t1ww)

[GPIOs 15](#_fl11nr264b3z)

[I/O Isolator 16](#_iub5wh1ala01)

[Debugging 18](#_v3yh6po0mhd)

[PCB Layout 24](#_dt94o3j5rqkh)

[**Appendix A 24**](#_8837mon8bsue)

# **References**

[1] CCP- Subsystems - POWER - Design - SC-ODIN-100, Spacecraft Electronics - SPEAR-M7 - Rev A [internal document on Google Drive], “SCH-SCSD20-EPS-001-A”. Available from: <https://drive.google.com/file/d/1Hjo98G_CQyjNqB3IPMDdYu_njFDT6Aj3/view?usp=sharing>

[2] Marki - RFTOOLS [website], “LC Filter Design Tool”. Website link: <https://rf-tools.com/lc-filter/>

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

[4] CUBICS - ELEC - Learning Resources - Learning Package [internal document on Google Drive], “Peripherals Design”. Available from: <https://docs.google.com/document/d/1Y1tGknWdWDa7ZUS9X4p9ahikzXoImTVWvahNcEV4DlE/edit?usp=sharing>

[5] Microchip datasheet, [PDF document], “SAM E70/S70/V70/V71 Data Sheet”. Link to document: <https://ww1.microchip.com/downloads/aemDocuments/documents/MCU32/ProductDocuments/DataSheets/SAM-E70-S70-V70-V71-Family-Data-Sheet-DS60001527.pdf>

[6] CUBICS - ELEC - Design [internal document on Google Drive], “Breakout Boards”. Available from: <https://drive.google.com/drive/folders/1IolXB_eUfjbBWtSN64kAr6lkOlDXZvvk?usp=share_link>

[7] 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=share_link>

[8] CUBICS - ELEC - Learning Resources - External Resources [internal document on Google Drive], “PCB Design Tutorial by Dave Jones”. Available from:

<https://drive.google.com/file/d/1sQq1nrMSX9F_TmR0TZrsZLF_wutw7tkj/view?usp=share_link>

[9] Analog Devices [Website], “Crystal Considerations with Maxim Real-Time Clocks (RTCs)”. Available froom:

<https://www.analog.com/en/design-notes/crystal-considerations-with-maxim-realtime-clocks-rtcs.html>

# **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 |
| ESD | Electrostatic Discharge |
| GPIO | General Purpose Input/Output |
| LED | Light Emitting Diode |
| LVDS | Low-Voltage Differential Signaling |
| MCU | Micro Controller Unit |
| 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**

MCU design is one of the more complex aspects of designing PCBs. Margins for errors are much smaller and proper layout rules are crucial. With that said, if done following a precise set of rules and conventions, MCU design can be fairly simple to adapt from one design to another as the procedure stays more or less the same for most MCU that will be used in FREYR and even future projects. This document will go in depth on those rules and how they can be applied across virtually any MCU design, from proper power and grounding to supporting hardware and ending on layout rules given how important they are.

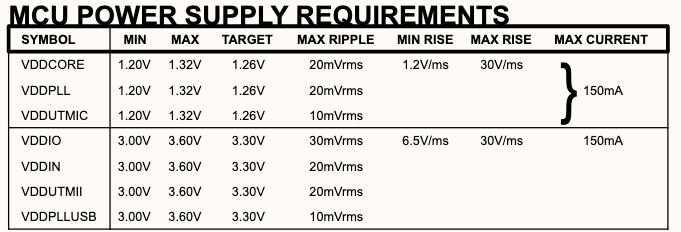
Throughout this document, two MCU in particular will be used as examples: the ATMEGAS64M1-MA-HP and the ATSAMV71Q21B-AAB. This will offer both extremes in terms of QFP package, with the ATmega having 32 pins and the ATSAM having 144. This is particularly relevant for SC-FREYR since the new architecture will utilize both a 144 pin MCU and many 32 pin MCUs. Knowing how to properly implement both of these will be crucial to any redesign from SC-ODIN.

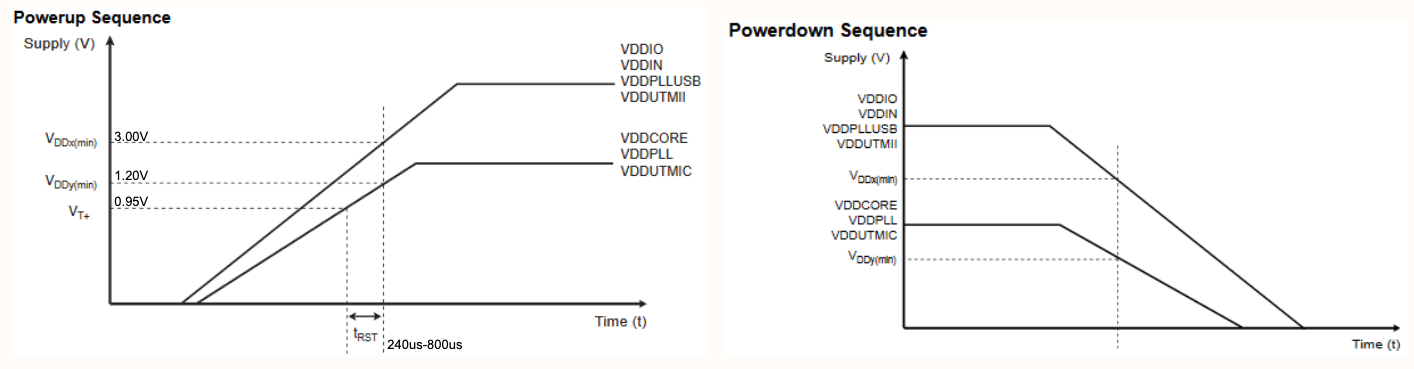
# **MCU**

## Power

MCU are one of the more complicated devices to power. They are generally very sensitive to noise, spikes and any other disturbances on the power lines. They even have detailed power down and power up sequences that need to be supplied by the chosen converter. Proper grounding techniques will also be explored in this section as it often is a major source of noise when improperly done. Filtering will also be explored in this section, although not in depth given the complexity that it can reach. This section will particularly look at the implementation of the ATSAM on the SPEAR-M7, the flight computer of SC-ODIN.

The first step in any MCU design, or any design in general, is to take a look at the datasheet and extract the relevant information for each step of the implementation. For power, it is very good practice to show on the schematic design the power requirements of the MCU, the power up sequence and the power down sequence. This can be seen on page three of reference [1] and on Figure 1 & 2 and should be present in any MCU design sheet for spacecraft electronics i.e. if it is going to space. This ensures that the chosen converter can satisfy the requirements set by the MCU, and also makes it easier to evaluate power consumption, noise generation and software requirements.

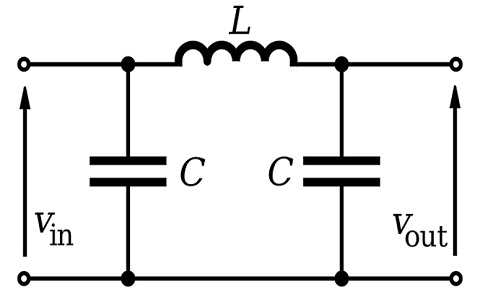
**Figure 1: ATSAMV71 power requirements**

**Figure 2: ATSAMV71 power up and power down sequence**

EMI Filter

Before implementing any power regulation, it is necessary to ensure that the overall power input of the board is free of any noise from other parts of the electrical system. The way to ensure this is by implementing a power filter at the board power input. Since power filters can get very complex, only one type of filter will be explored in this section: the PI filter. It is a simple, low component count and bidirectional filter, which means it will filter the input as well as the output to prevent a design from inducing noise into the rest of the system. This filter is only really required for MCUs and RF designs, since they tend to be more noise sensitive and noise generating. Note that this is also only necessary for designs that are a part of a larger system, not for breakout board designs or isolated designs.

A basic PI filter requires three elements: one inductor and two capacitors. There are two main types of PI filter: the Chebyshev and the Butterworth. Although the Chebyshev tends to have steeper decline in the frequency domain near the cutoff frequency, it trades it off for a generally worse time domain response compared to the Butterworth, which is why the latter is the recommended design for MCU filters. They are typically arranged in the configuration seen on Figure 3. The variation in the inductor and capacitor values will dictate how the filter acts from Vin to Vout and vice versa. The convenient thing about this configuration is that if both capacitors have the same value, the symmetry makes it that the backwards frequency response will be the same as the forwards response.



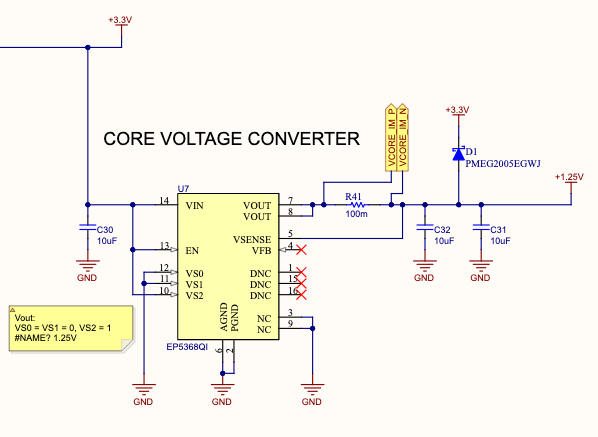
**Figure 3: PI filter basic configuration**

Figuring out the right values for any given application is the more challenging part of such filters, especially when forming RLC networks that can have high resonance. Fortunately, websites such as reference [2] can greatly help in finding those values. On this particular website, the PI filter is the equivalent of a lowpass butterworth 3rd order filter with shunt first topology. The desired cutoff frequency, input impedance and output impedance can be set to find the inductor and capacitor values. The impedance values are known from the specific design configuration that is being implemented, and the required cutoff frequency will be defined by the chosen MCU. At first, this looks fairly simple; evaluate the design to find the impedance values and check the data sheet for recommended cutoff frequency. Unfortunately, these filters are extremely sensitive to any type of parasitics, whether they are inductive, resistive or capacitive. This means that not only will the chosen inductance and capacitance value have a big impact on the frequency response, but the specific chosen components will as well. This is because all capacitors and inductors have parasitic effects, and some have more than others. Appendix A goes in depth on the design process of these power line filters and how to account for parasitics.

Power converter

Now that the power requirements are well defined and readily available on the schematic and that the input line is filtered, a power converter that matches the requirements can be chosen. The first choice that will be encountered when looking for a power converter is to choose between a linear regulator or a switching converter (also called buck converter). Reference [3] goes in depth about the difference between both, but for MCUs it is recommended to use buck converters. The next parameter to look into is voltage conversion. It needs to be able to take the board's power input, usually 5 or 3.3, and convert it to the required voltage. Although this may differ depending on the chosen MCU, the ATSAM requires a 3.3V and a 1.25V input. This means that if no 3.3V rail is available on the current design, two converters will need to be implemented. Another parameter that needs to be taken into consideration is the ripple factor of the converter. Going back to Figure 1, the datasheet will specify the maximum allowed ripple factor and the chosen converter needs to meet this requirement. Lastly, the selected DC/DC converter needs to be able to supply the required current from the MCU. This particular MCU, as seen in Figure 1, requires a total of 300 mA at maximum operation. It is good practice to choose a converter that can supply more than the required current to account for power loss and to reduce the margin of error.

For the SPEAR-M7 design, the selected converter was the EP5368QI. This step down DC-DC converter (another name for power converter) offers a low ripple factor, adjustable voltage output with a supply current of 600 mA. Three pins allow to set the voltage output of the converter, making it compatible for 3.3 and 1.25 volt conversion. On top of the required features, it also has short circuit protection, under voltage lockout and thermal shutdown. Figure 4 shows the implementation of this chip to convert the 3.3V input (input of the entire board) to 1.25V rail.



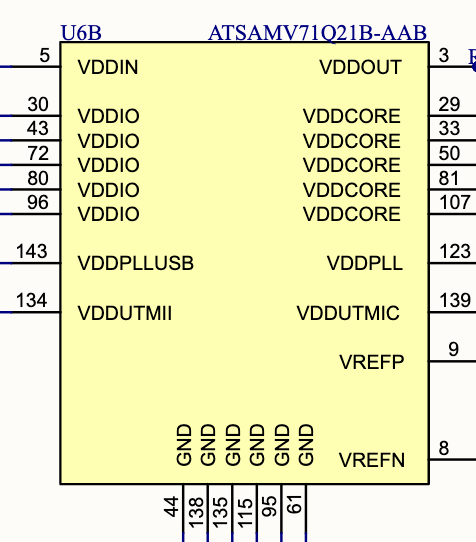
**Figure 4: EP5368QI implementation on the SPEAR-M7**

Providing the equation for voltage selection, like done in Figure 4 (pin VS0/1/2), is always good practice and makes it easier for the design reviewer to validate the design. It is important to note that current sensing is implemented using a shunt resistor, which should always be done for self sustaining designs. Refer the ADC section of reference [4] to understand current sensing. Another important thing to note is the presence of big decoupling capacitors, which are crucial to maintain low noise power rails. Lastly, D1 is a schottky diode which will protect the 1.25V line in case of a massive inductive spike that can occur from switching converters (read on schottky diodes).

One thing to note on page three of reference [2] is the VDDOUT pin on the ATSAM power block. The datasheet specifies that the MCU has an integrated voltage converter to provide the 1.25V rail. Although this can be used for ground equipment design, it is recommended to implement an external converter for space design for redundancy purposes. MCUs are very sensitive to radiation and even if the particular MCU chosen is radiation resilient, the internal converter is still subject to a higher risk of failure. External converters also tend to have better stability and noise reduction circuitry. It is still good practice to place a 0R DNP resistor in series to allow the use of the internal converter during testing or for other purposes. With that said, make sure the DNP is visible and listed in the 0R properties to avoid assembly errors.

Input Filtering

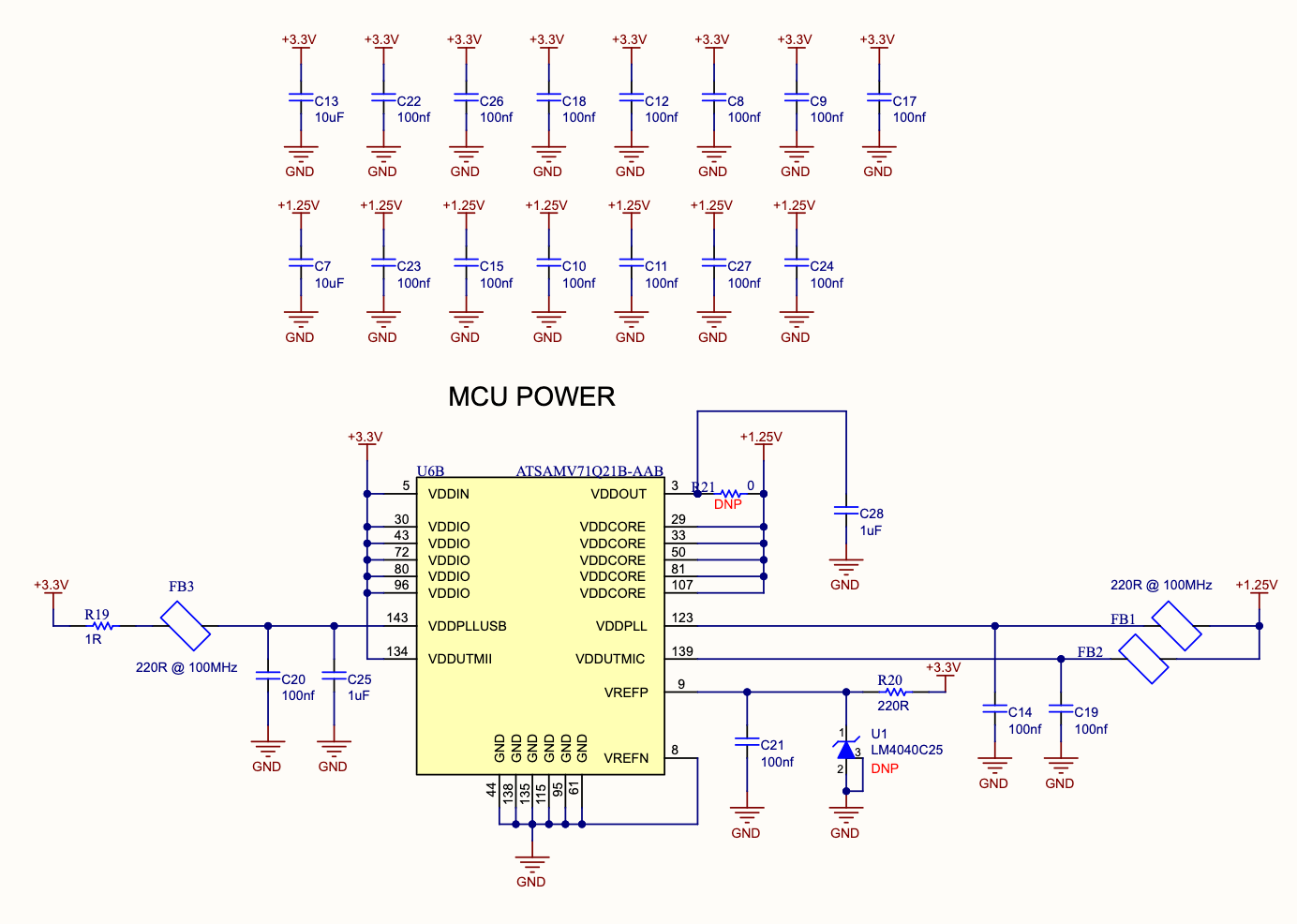
One last filtering stage is needed for MCU power input pins. As seen on previous designs, one 100nF capacitor should be added to every power input pin for proper decoupling. Some specific pins will need added filtering, most of the time in the form of an extra capacitor with a different value or an in-line ferrite bead. Ferrite beads are used a lot in Spacecraft’s design for their ease of implementation and great noise reduction properties (highly recommended to read up on ferrite if unfamiliar with the component). The pins that will require the extra filtering can be found in the datasheet. For the ATSAM chip, the datasheet is nearly 2000 pages long, so it might require some time to locate the relevant information. The data sheet of the ATSAM can be found at reference [5]. This is where the information about the recommended filtering can be found. More precisely, page 1873 and the following pages contain recommended capacitor, inductor and ferrite bead values for decoupling and filtering on the specific power pins. First, let’s look at the power block schematic symbol of the chip.



**Figure 5: ATSAMV71 power block schematic diagram**

The data sheet specifies the following connections: VDDCORE, VDDPLL and VDDUTMIC need to be powered by the 1.25V supply and VDDIN, VDDIO, VDDPLLUSB and VDDUTMII need to be powered by the 3.3V supply. VREFP and VREFN are used for the internal ADC, which will be looked into in another section of this document. VDDOUT is the internal power converter output, and should follow the previously mentioned implementation.

On page 1873 of the data sheet, the recommended filtering for each of these power pins is shown. VDDIO, VDDUTMII and VDDCORE all require a single 100nF capacitor on each pin. VDDPLLUSB, VDDPLL and VDDUTMIC all three require the 100nF capacitor, and a ferrite bead or inductor resistor combo in line with the input pin. For ease of component sourcing, an identical ferrite bead can be used on all three inputs. The data sheet recommends a 470R @ 100 MHz ferrite bead, but similar values can also be used. Lastly, an extra capacitor is required on the VDDPLLUSB line, either the recommended 4.7uF or other similar value used in the design. Figure 6 shows the final implementation of the power block for the ATSAMV71 chip. All some of the values present are based on part sourcing constraints, like the ferrite bead value.



**Figure 6: ATSAMV71 power block implementation**

Important to note that for ease of reading, all 100nF decoupling capacitors are grouped away to not cram the schematic block. The will be assigned to a particular pin when doing the PCB layout. It is also recommended to have a 10uF bulk capacitor for every voltage net (one for 3.3, one for 1.8 and etc). One last thing to note is the presence of the 0 ohm DNP resistor on VDDOUT, which refers to the DNP resistor mentioned at the end of page 11.

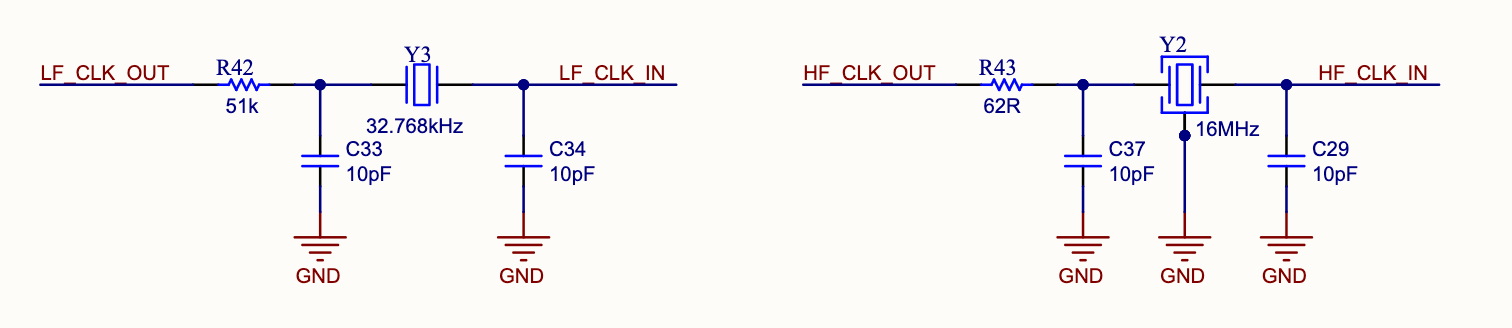
To reiterate, each MCU will have specific power considerations, noise sensitivity and implementation recommendations. Small MCU are generally really simple to implement: some of them will only require a single 100nF capacitor and a ferrite bead between the main power and its power input pin. The goal of this section was to show a proper implementation of a large MCU, and the rules and conventions used can be applied to most designs. Adjustments need to be made following the manufacturers recommendations, but it will rarely be significant.

## 

## Crystals

Crystals are an essential part of MCU design, since every single MCU requires either a crystal oscillator or an external clock generator. Some smaller package MCUs will even have an internal oscillator, but it is generally good practice to implement an external crystal if the pin count allows it. These oscillators can have multiple uses, but their main use is to generate stable and accurate clock signals. They are also commonly used to count time in RTCs.

Crystals are usually fairly simple to implement. MCUs have dedicated pins to connect the oscillator to, often marked with the prefixes X or OSC (might differ depending on the chosen MCU). Fortunately, crystals are bidirectional and can simply be connected in any orientation. Bigger MCU will have two sets of dedicated pins, one for high frequency oscillator and one for low frequency (usually marked HF and LF). The only element required when connecting a crystal is two capacitors going to ground on both leads of the crystal. This value will depend on the frequency of the crystal. Figure 7 shows the two crystals used on the ATSAMV71 and their implementation. Resistors may be added in series following the datasheet recommendation for max ESR and internal equivalent capacitance.



**Figure 7: Crystal Oscillators implementation**

The most challenging aspect of crystal oscillators is their PCB layout considerations, which will be discussed in another section of this document.

## 

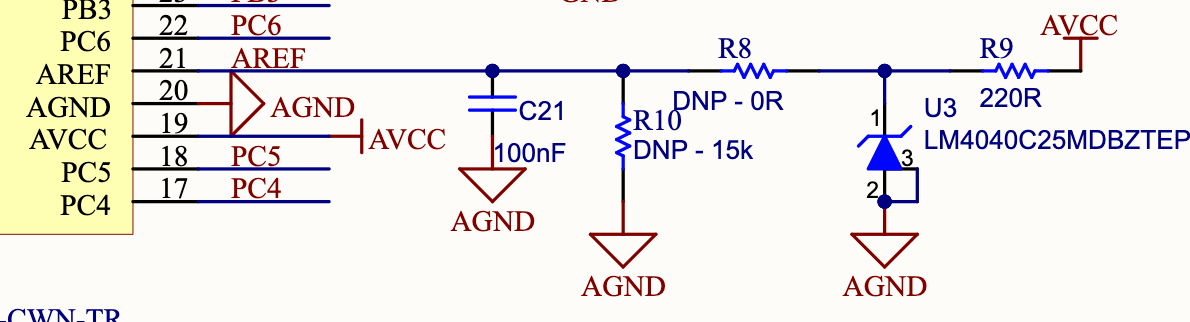
## Internal Peripherals

Many MCUs have embedded peripherals such as ADCs, RTCs, DACs and more. How many are included will often depend on the size of the package and the complexity of the MCU. They can be accessed through specific pins or even regular GPIO if the architecture allows it. This section will look at different types of internal peripherals, the advantages and disadvantages of them and how to properly implement them.

ADC

Almost every MCU has an internal ADC that is available to use. Depending on the size of the MCU, it might be single channel or multi-channel and the precision will vary with MCU complexity. All of the information concerning the internal ADC can be found in the MCUs datasheet. These have some advantages: they are easy to implement and don’t require any communication lines to function, unlike external ADCs. The disadvantage of them is that the number of channels is often limited and they can only be used to measure analog voltage close to the MCU itself, otherwise too much noise will be introduced.

Their implementation is usually very simple and similar to external ADCs: the MCU will have an analog VCC port and an analog GND port, which should both be kept separate from the main power and ground of the board following the design practice mentioned in reference [4] page 8. The MCUs datasheet will then specify which I/O can be used as an analog read pin. Large MCUs will offer multiple input channels and allow the user to access them through any GPIO, making the implementation really convenient. On the other hand, smaller MCU will have a restricted number of channels and might only be accessible through certain I/Os. If the number of I/O is too low for a required design, an analog switch or an external ADC should be considered. Last aspect of internal ADCs to consider is the voltage reference. Some MCUs will have internal selectable references that can be used for A/D operations, but some MCUs will also require an external voltage reference to be connected to a port, commonly named AREF (analog reference). If an external reference is required, components such as the LM4040 are simple to implement and will supply the appropriate voltage reference. Figure 8 shows the implementation of an internal ADC on the ATmega64 MCU which will be used throughout the design of SC-FREYR.



**Figure 8: implementation of the internal ADC of the ATmega64**

It can be seen that a separate VCC and GND ports are assigned with the internal ADC. Additionally, an external reference needs to be connected on the AREF pin. This external reference can be easily attained by using a component like the LM4040. Simply connect the anode to AVCC and the cathode to ground, and the component will automatically generate a reference voltage. R8 is there to allow the reference to be disconnected if necessary, and if it is not populated R10 should be placed to ground the AREF pin. C21 is the usual decoupling capacitor.

RTC

Depending on the complexity of the chosen MCU, an internal RTC may be available. It is generally not recommended to use for space applications simply because of the reliability issue. With that said, it can be used in pairs with an external clock and compared to mitigate the effect of clock drift.

## GPIOs

This section will glance over the different functionalities and features of MCU pins. This section will specifically look at different features that MCU pins can have and how they can be used for different designs.

Tri-State Logic

Tri-state logic, or three state logic, is a term often used when describing the functionality of a specific GPIO or set of GPIOs. It specifies that the GPIO has three possible states: high, low or high impedance. The high and low functionality are typical of an output pin, since it is able to drive the line high or low to control what it is connected to. The high impedance mode is the “input” mode. The fact that it is high impedance allows the pin to not interfere with the circuitry connected to it, but still allows it to sense a high or a low.

This brings a massive advantage for implementing complex digital design: signals that need to be controlled or be fed to the MCU can be connected to virtually any GPIO, since they can be set as inputs or outputs through software configuration. Only a few pins on an MCU are reserved for specific functions, which makes the overall integration much more simple. If an MCU does not specify tri state logic on a specific GPIO or set of GPIOs, these can not be set by software and will either be an input or an output, not both. This is important to look for since reversing the polarity of the input could lead to damage to the MCU.

Internal Peripherals Access

As complexity and size of MCUs increases, the number of features and internal peripherals will increase and can become very difficult to plan around if peripherals are only accessible through specific pins. To solve this issue, newer MCUs started offering internal peripheral access on virtually any GPIO, making the organization much easier and much more flexible. The chosen MCU obviously has to have that feature mentioned, but larger MCUs usually have this feature.

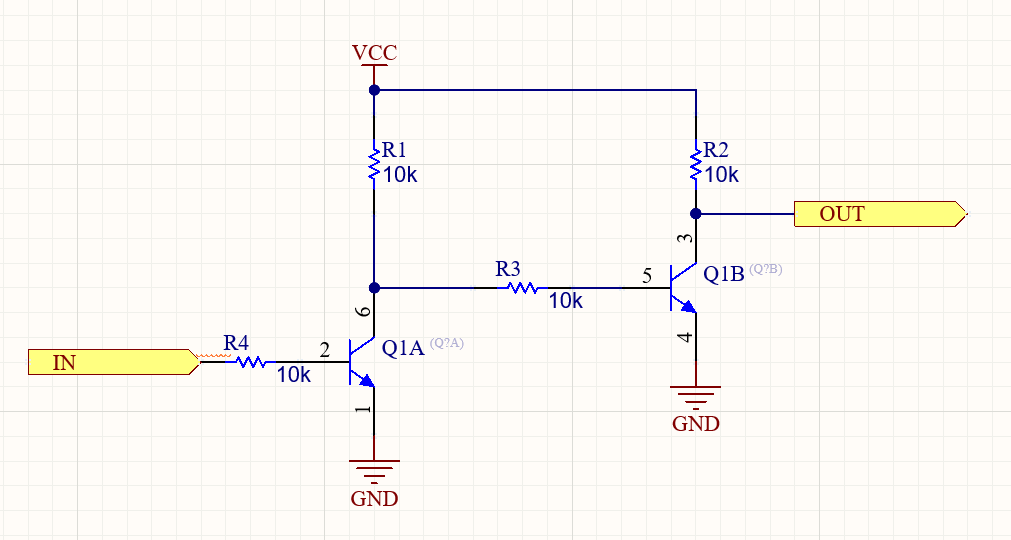
A great example of how this can be used is how the PIC16F15224T-I/MG is implemented on the EPS intro task (rev 4.0). The task requires the member to design a board around this very small MCU. One thing that the design requires is to have voltage sensing over a potentiometer and current sensing on the voltage line. The issue is that the PIC16F15224T-I/MG only has one internal ADC channel, so the members have to find a way around this issue. One of the solutions involves understanding internal peripheral access; the member can simply connect the current sense and voltage sense to two different GPIOs that have access to the internal ADC and simply read one at a time and switch the access to read the second value. This is much simpler than using an analog switch controlled by another GPIO.

Internal peripheral access also often allows different communication protocols to be accessed through different ports, which makes implementation with other components much easier.

## 

## I/O Isolator

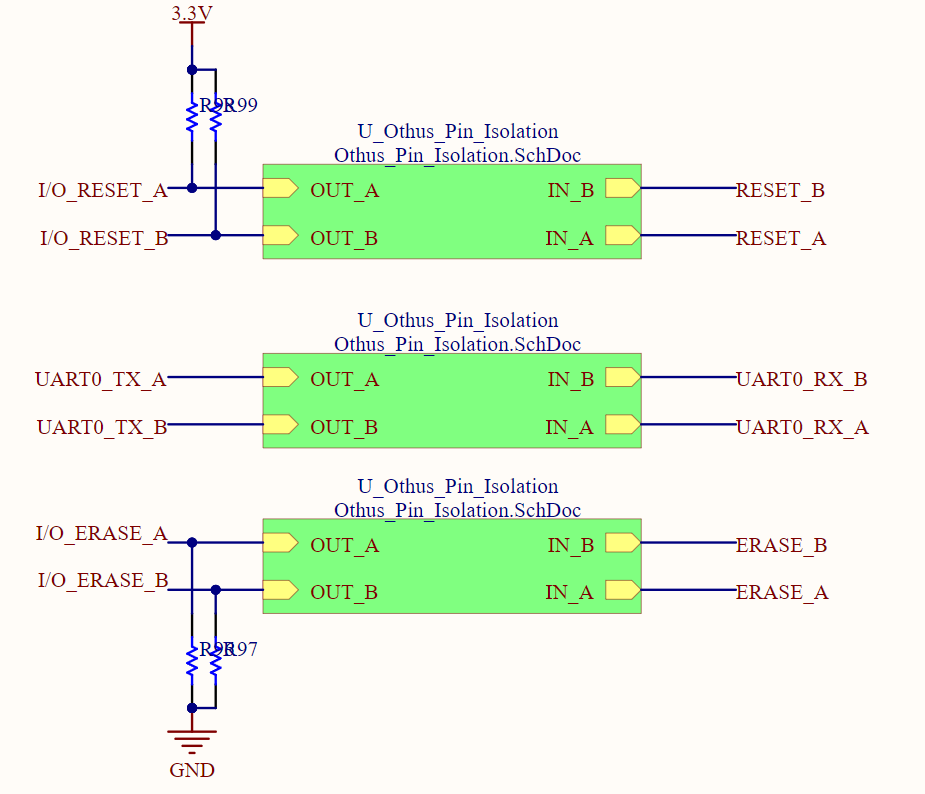
An I/O isolator is an isolation circuit that is used when connecting multiple MCU I/Os together. This is usually done when implementing redundant MCUs since they are required to communicate with each other, typically through UART ports and GPIOs to control reset and erase commands. Redundancy can only truly be achieved if both MCUs are isolated from each other’s faults. If two GPIOs or UART ports are directly connected together, a major failure from one MCU could destroy the second MCU. To avoid this, it is necessary to use the pin isolator shown in figure 9.



**Figure 9: Pin Isolator Circuit**

The way this circuit works is fairly simple. Instead of driving the output directly from the input, the input will bias or not Q1A, which will in term bias Q1B to the required output. If IN is low, Q1A is in the cutoff region, which allows Q1B to be directly biased by VCC. This directly connects OUT to ground, copying the low from IN to OUT. When IN goes high, Q1A is now in forward active mode, which effectively grounds the base of Q1B, putting it in cutoff mode. With Q1B in cutoff, OUT is directly connected to VCC and is therefore high.

One important consideration to take into account when using this pin isolator is that leaving the input floating will create an unknown state on the output. This is only an issue for important signals such as a RESET and ERASE control signal. Leaving them in an unknown state could trigger a RESET or ERASE even. To prevent this, pullup and pulldown resistors should be placed **on the input of the isolator, not the output.** Figure 10 shows how this was done on the CDH board of SC-ODIN, which serves as the interface between the two main flight computers. Note that they are used as sheet symbols in this implementation, but the circuit stays the same.

****

**Figure 10: Pin Isolator with Pullup and Pulldown Resistors**

## 

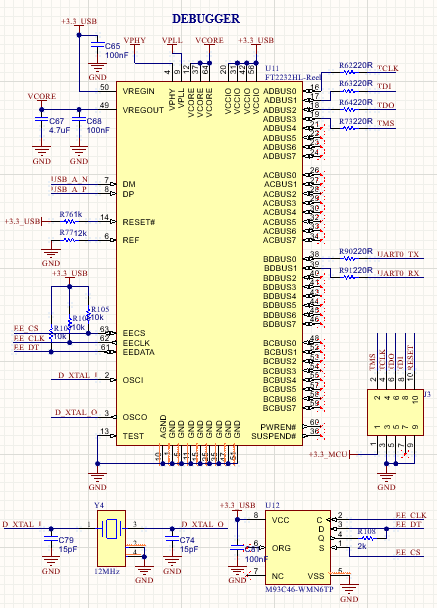
## Debugging

MCU debugging is a crucial part of MCU Design since it allows for precise testing of the hardware implementation by the software team. Debugging is usually handled by a completely separate MCU whose sole purpose is to control the debug ports of the main MCU being tested. On breakout board designs, these debugging MCUs are integrated on the same PCB as the main MCU to allow simple and reliable connection between them. All that is required to do is to provide a connection to the debug MCU, and a connection to the main MCU. On final satellite PCB designs that integrate MCUs, the debug MCU should not be put on the PCB since it takes a lot of space that will not be used during normal operation of the satellite. For this reason, the debug ports of the MCUs should be broken out so an external debugger can connect and do the testing. This solution takes much less space on an already small PCB and will not leave unused hardware on the PCB. This section will look at both internal and external debugger implementations and general design rules surrounding both.

First, it is important to know what protocols are used for MCU debugging. There are two main protocols used, with each their own implementation. A third option is also available which integrates both on the same physical connections. The two main protocols are JTAG and SWD. Without going in depth into both, the main difference is the number of physical wires required for them. JTAG uses four to six while SWD uses two. Although it would make sense to use less cables, JTAG will be the preferred choice for SC-FREYR’s design because of the availability of the debugging chips and there is a lot of documentation available for this chip. The required pins are TCLK (clock), TDI (data input), TDO (data output) and TMS (test mode select). If the debugging is implemented externally, a mutual ground will need to be connected as well as a voltage reference pin.

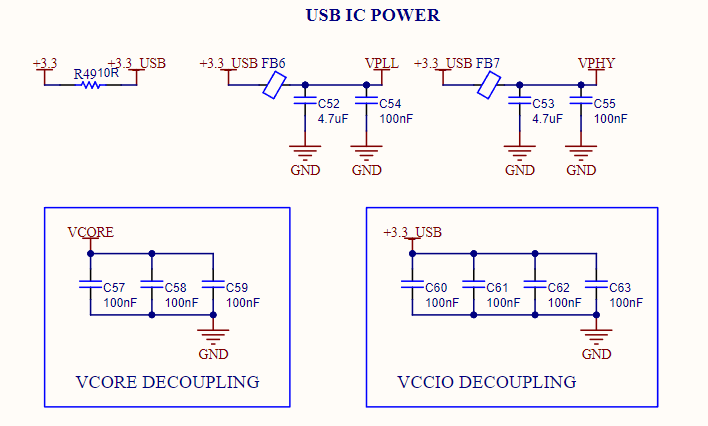
Internal Debugger Implementation

The first implementation that will be shown in this section is the internal debugger implementation. This means that the debugging MCU will be on the same PCB as the main MCU it is trying to debug. A great example of this is the CDH breakout board developed for SC-FREYR. The main goal of this design is to make a breakout board that will allow the software team to test and debug the MCU chosen for the main flight computer. This will ease the development and testing process of the flight software and allow them to test out the interaction with a variety of peripherals. The debugging chip used on this design is the FT2232HL-Reel and the main MCU is the SAMV71Q21RT This section will look at the implementation of the debugging MCU as well as the connection between it and the main MCU. The main MCU’s implementation can be found in the sections above. The implementation of the FT2232HL-Reel can be seen in figure 11.

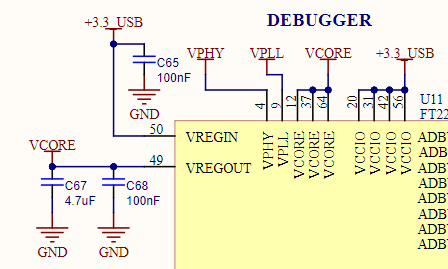


**Figure 11: Implementation of the FT2232HL-Reel**

This schematic obviously contains a lot of different parts, and this next subsection will look at each of them at a time. First, the power must be set properly. It is not as complicated as the power block of the SAMV71, but it is still crucial for it to be done correctly. Figure 12 shows the main decoupling and power filter components for the MCU, while figure 13 shows how to connect them.



**Figure 12: FT2232HL-Reel Decoupling Block**

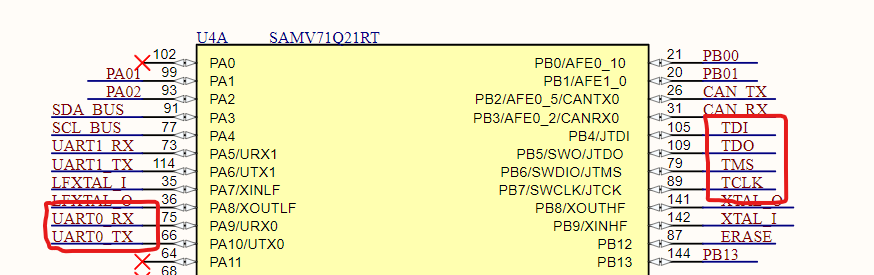


**Figure 13: FT2232HL-Reel Power Connections**

Most of the decoupling terminology is taken directly from the datasheet (page 51 of the [datasheet](https://datasheet.ciiva.com/1787/ds-ft2232h-1787543.pdf?src-supplier=Digi-Key)). Since this breakout board will be used for testing purposes and not for space applications, using the internal voltage regulator makes the implementation much simpler and is used in this case. As always, ensure to have one 100nF capacitor per power pin, and use the standard ferrite bead instead of the suggested inductors.

Back to the implementation seen on figure 11, the whole point of this component is to provide MCU debugging through a simple USB port on the breakout board. For this reason, a USB connector should be implemented following the recommendation found in reference [7] page 18. The two USB nets should then be connected to the dedicated USB pins DM (data minus) and DP (data plus). The RESET# should be pulled up high through a 1K ohm since it won’t be necessary to reset the MCU. Per the datasheet instructions, REF should be grounded through a 12K ohm resistor and the three EE pins should be connected to an EEPROM per the datasheet (page 55 of [datasheet](https://datasheet.ciiva.com/1787/ds-ft2232h-1787543.pdf?src-supplier=Digi-Key)). OSCI and OSCO are the typical crystal oscillator input and output pins that should be connected to a 12 MHz crystal with external capacitor on both pins. Lastly, TEST can simply be grounded and PWREN# and SUSPEND# can be left floating.

Now that the MCU is powered on and properly set up, the last step is to connect it to the main MCU using the correct connections. As mentioned earlier, the preferred debugging protocol that will be used through the course of SC-FREYR is JTAG. The four required connections can be seen at the top right of figure 11. TCLK is the clock line, TDI is the data input line, TDO is the data output line and TMS is the test mode select line. Since this implementation is done on the same PCB as the MCU it is trying to debug, there is no need for an additional ground line and an additional VREF line. One thing to note is the presence of UART ports. Although not necessary, it makes the communication between both MCU much simpler for our CDH team and should be connected unless space restrictions prevent it. The connections on the main MCU can be seen on figure 14.



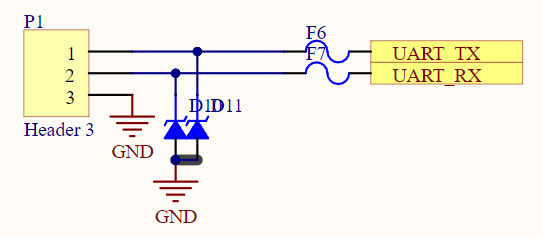
**Figure 14: UART and JTAG Connections on the SAMV71**

It is good design practice to break out the JTAG ports on a header to allow for external debugging in case the internal chip fails. This can be seen on figure 11 through the J3 connector.

External Debugger Implementation

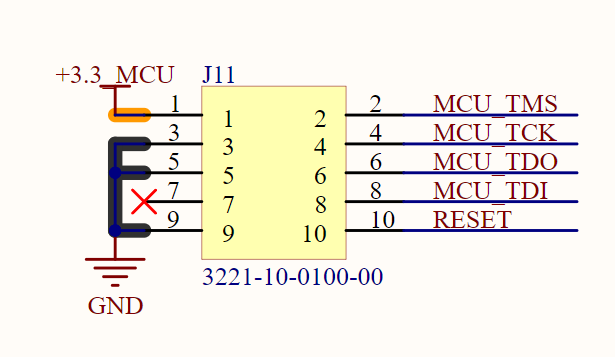
The external debugger implementation is very similar to the internal implementation, with the difference that two different PCBs are required to be connected to each other. This section will not look at the design of the external debugger PCB since this will be thoroughly described and explained in the user manual for the MCU External Debugger design (SC-FREYR-EPS-709). Rather, this section will look at what needs to be broken out from the main MCU to allow the external debugger to connect to it. This will be valid for all EPS-100 designs that will integrate an MCU.

There are two common protocols used for debugging and running tests are JTAG and UART (see figure [7]). For UART, what needs to be broken out is pretty straight forward: the RX and TX lines with one ground. This can be seen in figure x. The fuse and ESD diodes are not required for flight components, only for ground equipment. It is also good to note that virtually any header or 3 pin connector will work for UART since they will then be manually wired using jumper wires. No specific connection layout is required.

****

**Figure x: UART Header Configuration**

JTAG on the other hand requires a specific connector (or at least for Spacecraft equipment) and has a predefined pinout that needs to be followed across all systems to ensure compatibility. The connector used is the CNC Tech 3221-10-0100-00 or its right angle variant, the 3221-10-0200-00. In both cases, the pinout needs to follow the one shown in figure x. It is important to note that this connector is not bidirectional and will only fit in this specific direction.

****

**Figure x: JTAG Connector Pinout**

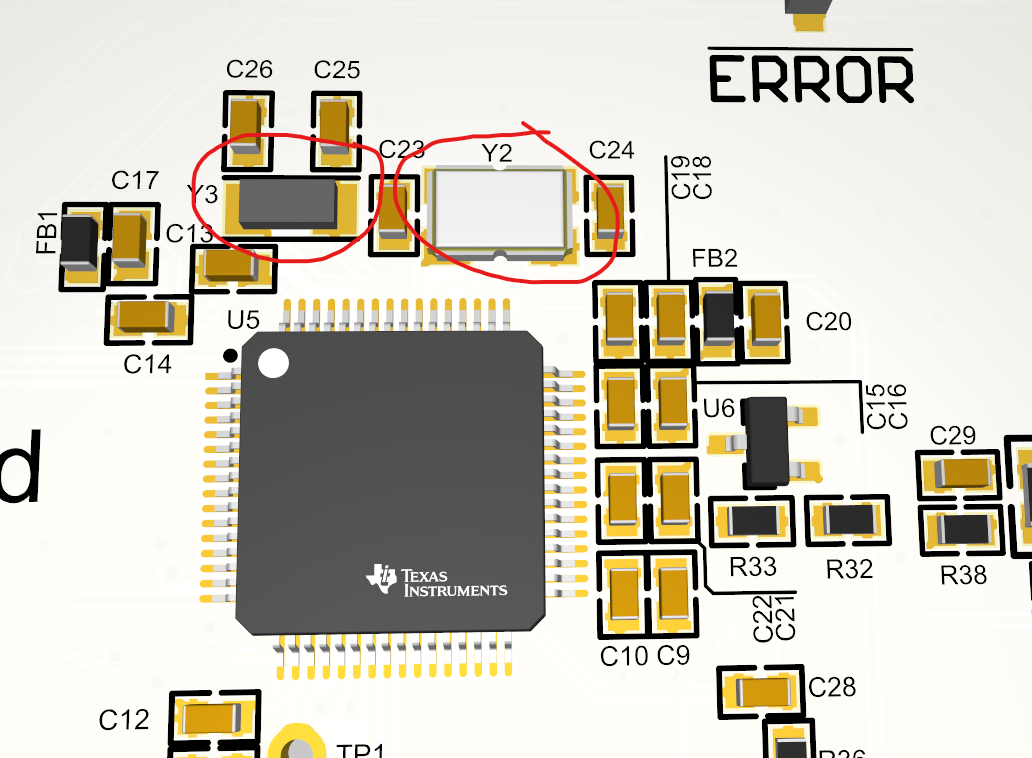
Again, it is important to follow this exact pinout since the External Debugger (709 board) will interact with all the designs using this specific pinout.

## 

## PCB Layout

Before beginning the PCB layout of an MCU, it is highly recommended to read the tutorial listed in reference [8] as it goes over a lot of very important concepts in PCB layout, some of which apply to MCU design. This section will mainly focus on the MCU specific design requirements and best practices.

The first thing that should be considered when laying out an MCU is the crystal. As mentioned before, it is the most critical part of MCU layout and needs to be kept as close as possible to the MCU, which is why it should be the first component to be placed. The only thing that needs to be considered before placing the crystal down is to ensure that its placement won’t interfere with important data lines such as USB and other high speed protocols. Figure x shows the placement of the two crystals required for the SC-FREYR 16-bit MCU.



**Figure x: Crystal placement on the MSP430FR5989IPMR**

This layout ensures that the trace length between the crystals and the MCU is kept a minimum and that both load capacitors are as close as possible to the crystals themselves.

Crystals are very intricate components and require a lot more considerations than mentioned in this document if the goal is to implement them as well as possible. More in depth resources can be found under reference [9]

# 

# **Appendix A**

There are multiple levels of accuracy to which these pi-filters can be designed to. Although it might not be necessary to go down all the way to extreme precision, the more accurate the design is the safer it will be.

The first step to ensure proper response of the pi-filter is to simulate it using LTspice, a free circuit simulation software commonly used. The first step will obviously be to download said software ([download link](https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html)).

# 