

**Goal:**

You and your team, and for that matter, the entire ME218c class, have been contracted by an intergalactic corporation to respond to a distress signal. As it turns out, some miners have been trapped underneath the surface of a faraway planet after an underground explosion, and the class has been charged with developing a Basic All-Terrain Planetary Observation and Detection System (BATPODS) consisting of 14 planetary vehicles, which must carry remote camera payloads and interact with the environment, and 14 orbital controllers, which must transfer control of the vehicles between them as they orbit and move in and out of communications range.

**Purpose:**

The underlying purpose of this project is to provide you with an opportunity to gain experience in integrating all that you have learned in the ME218 course sequence, with an emphasis on the new material in ME218c.

**The Task:**

Design and build a tele-operated Basic All-Terrain Planetary Observation and Detection System (BATPODS) consisting of a Remotely Operated Aid for Mine Evacuation and Recovery (ROAMER) and a companion Personal Operation Datalink (POD). Each ROAMER must carry an SPDL-supplied Solar-Powered Environmental Camera Interface with Assistive Lamp (SPECIAL) to aid the operator of the POD in navigation. Groups of ROAMERS will co-operate in a simulated alien terrain to rescue a group of trapped miners.

---

**Specifications**

**General:**

1. Each team will construct a ROAMER and a POD.
2. The ROAMERS are devices capable of navigating an Alien Environment to rescue stranded miners after an underground explosion.
3. The PODs are the wireless remote controllers for the ROAMERS.

**Basic Game Play:**

4. A game round will be a collaboration among three ROAMERS to attempt to rescue a small group of miners trapped by an underground explosion. These mines have been developed by corporation more interested in production than in miner safety. As a result, up to date maps of the mines are not available.
5. The goal of the game is to locate the miners, release them from the mine and bring them to the extraction point.
6. The game will continue until either all of the miners have been rescued or all of the ROAMERS' remote cameras have lost their store of energy..

**The Planet:**

7. The planet is composed of a remote region into which three ROAMERS will be deployed.
8. The surface of the mine is mostly level with regions of both smooth and compressible substrate. Surface samples will be available real soon now.
9. There is a single shaft of natural sunlight at the Landing Zone (LZ).
10. The planet has frequent sand storms with wind-speeds in the mine approaching 30mph.
11. Steam from geothermal eruptions results in occasional regions of fog.
12. The mine is dark, which impairs the operators' vision through the remote cameras.

**The Mission:**

- 13. The team of ROAMERs must navigate into the mine, locate each airlock and manually pull off a hatch to gain access to the miners' survival chambers.
- 14. The ROAMERs must then drag each survival chamber to the extraction point in order to save the miners inside. Seems simple enough.

**The ROAMERs:**

- 15. Each ROAMER must be capable of moving under its own power within the mine (described above).
- 16. ROAMERs must be battery powered and operate without a tether.
- 17. Remote control of ROAMER functions must be achieved via a POD using the provided RF hardware (XBee24 modules).
- 18. ROAMERs must incorporate an easily accessible switch that disables all moving systems.
- 19. Each ROAMER should be equipped with a gripper to enable it to pull off hatches and to grab the miners' survival chambers. The handles for the hatches and survival chambers are shown in Fig. 1 & Fig. 2.
- 20. The gripper may only be actuated and de-actuated in response to an explicit commands from the POD.
- 21. Each ROAMER must carry an SPDL supplied Solar Powered Environmental Camera Interface with Assistive Lamp (SPECIAL). The lens of the SPECIAL must be mounted at a height of 10" off the surface of the ground and extend 0.25 to 0.5" beyond the perimeter of the ROAMER for clear video transmission.
- 22. In order to fit into the drop-ship cargo hold for transport to the mine, the ROAMER must fit into a volume of no more than 12"x12"x16"H. ROAMERs not conforming to this size envelope will be instead dropped by orbital insertion.
- 23. The only thing allowed to extend beyond the perimeter is a gripper for opening hatches and dragging survival chambers. The extension of the gripper beyond the perimeter is limited to 6"L x 12"W x 3"H.
- 24. ROAMERs must be stable in the presence of a 30mph wind.
- 25. The ROAMERs may not emit visible or IR radiation of any kind during operation.
- 26. Each ROAMER must have an easily accessible switch to designate it as ROAMER 1, 2 or 3.

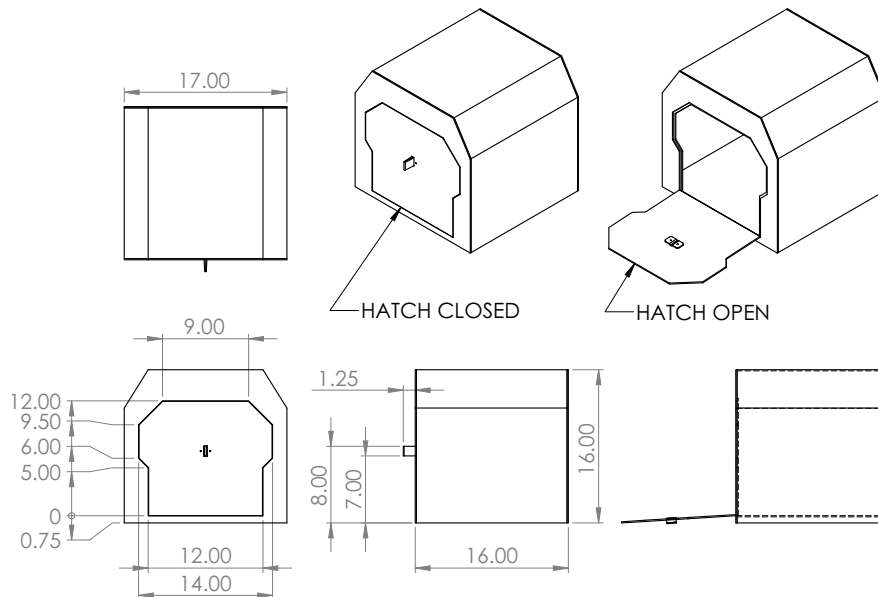


Fig 1 Airlock Hatch

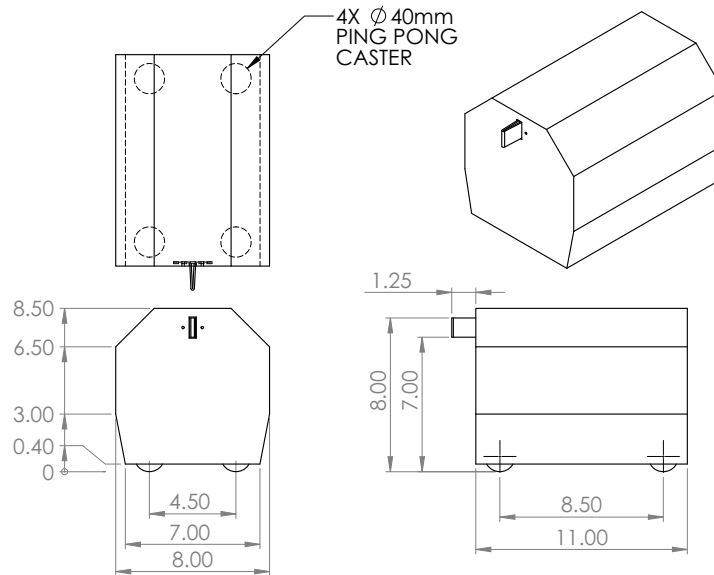


Fig 2 Survival Chamber

### The PODs:

27. Each team will design and construct a POD that will relay commands from a human operator to a ROAMER, and receive and display status information from the ROAMER.
28. The POD must be capable of displaying to the operator an indication of active communication with its associated ROAMER.
29. The POD must provide a method for the operator to designate which ROAMER (1, 2 or 3) the operator would like to control.
30. The POD must include an electro-mechanical display of the battery level of the SPECIAL on the ROAMER to which it is currently connected.
31. Each command to the ROAMER to actuate/de-actuate its gripper must be initiated by a user action at the POD.
32. PODs must be battery powered, and shall have sufficient battery capacity for at least 8 hours of continuous operation. The report should show documentation and calculations to support meeting this requirement.
33. PODs must be untethered, portable by one person and occupy a volume no larger than 2.5'Wx2.5'Dx5'H.
34. Input to the POD should involve at least 3 sensing modalities (e.g. position, force, audio, acceleration, etc.). Use of unusual interface methods is encouraged.
35. The actions required by the user of the POD to issue commands to the ROAMER should be inventive and interesting for the audience to watch. Use of actions that make the operator look and feel foolish is encouraged.
36. PODs should be intuitive to operate, and/or have sufficient visual instructions that a typical spectator (even a non-engineer) would be able to learn its controls within the time span of a single game round.

### The Solar Powered Environmental Camera Interface with Assistive Lamp (SPECIAL):

37. The SPECIAL is a self-contained subsystem that will provide the video link back to mission control as well as provide information to the ROAMER about the status of the SPECIAL batteries and control the assistive lamps. Detailed documentation on the interface to the SPECIAL is contained in a separate document.

38. The battery on the SPECIAL may be “charged” by placing it into the sunlight at the LZ.
39. The SPECIAL will have illumination LEDs that may be turned on & off by commands issued over the interface described in the SPECIAL documentation.
40. Turning on the illumination LEDs will drain the SPECIAL batteries faster.
41. As the SPECIAL batteries are depleted, first the external illumination will be lost; as it continues to discharge, the camera video will be degraded. When it is completely discharged, all video communications will be lost.
42. Communications with the SPECIAL must take place through a PIC16F690 programmed in assembly language.

### **Mission Control:**

43. Mission Control will be located in Thornton 110 where the PODs will be set up to control the ROAMERS operating in Thornton 210
44. At any point in time, all of the ROAMERS on the surface of the planet will be controlled by PODs on one of three orbiting platforms ( $\alpha, \lambda, \phi$ ).
45. Due to planetary rings, the platforms are in low, fast orbits and can maintain communications for only short periods of time. When one orbital platform gets out of range the PODs on the next orbital platform must assume control of the ROAMERS. Thus, 9 PODs are needed to maintain constant control of the 3 ROAMERS on the planetary surface for the duration of the mission.
46. The operator of a POD that wishes to control a particular ROAMER (1,2 or 3) must select that ROAMER using the POD and make a unique control action to initiate taking control of the ROAMER.
47. If a ROAMER receives a request for control while it is already under control, it will silently ignore the request.
48. Prior to the start of a mission, PODs will be assigned to one of the three orbital stations.

### **Radio Communications:**

49. Communications between the ROAMERS, and PODs, will take place over an SPDL-supplied 802.15.4 radio (Xbee24) using the Non-Beacon API mode of operation.
50. Any POD should be capable of controlling any ROAMER.
51. Once a mission begins, communication will take the form of bi-directional communications between a ROAMER and its bound POD.
52. Each ROAMER and POD will be assigned a unique ID in the form of the source address of each SPDL-supplied radio.
53. The details of the communications protocol will be defined and specified by a Communications Committee, which will consist of one member from each project team. The specification must be in a written form and with sufficient detail that someone sufficiently skilled in ME218 material could implement it.
54. In order to better balance the workload and learning among team members, each of the following tasks must be completed by a different member of the team: serve on the communications committee, implement communications on the ROAMER, and implement communications on the POD.
55. The class communications protocol must include a procedure for validation of communication between the ROAMER and POD. The PODs must provide a visual indication of when a functioning communications link between the ROAMER and POD exists.
56. The POD may issue commands to a ROAMER at a rate no greater than 5 Hz.
57. The ROAMER may issue responses to a POD at a rate no greater than 5 Hz.

**General Requirements:**

58. At a minimum, either the POD or the ROAMER must contain two actively communicating processors. There is no class imposed upper limit on the number of processors employed.
59. The microcontroller that interfaces with the SPECIAL must be programmed entirely in assembly language.
60. You are limited to an expenditure of **\$200.00/ team** for all materials and parts used in the construction of your project. Materials from the lab kit or the Cabinet Of Freedom do not count against the limit. All other items count at their fair market value.
61. A project logbook must be maintained for each group. An on-line blog is appropriate to meet this requirement as long as it is made available to the teaching staff for review. This book should reflect the current state of the project, planning for the future, results of meetings, designs as they evolve etc. The project logbook will be collected at irregular intervals for evaluation.
62. A report describing the technical details of the system will be required. The report should be of sufficient detail that a person skilled at the level of ME218c could understand, reproduce, and modify the design. The report must be in website format, and be suitable for posting on the SPDL site.
63. ROAMERs based substantially on purchased vehicle platforms are not allowed.
64. All projects must respect the spirit of the rules. If your team is considering something that **may** violate the spirit of the rules, you must consult a member of the teaching staff.

**Safety:**

65. Both the ROAMERs and the PODs should be safe, both to the user and the spectators.
66. No part of the ROAMER or POD may become ballistic.
67. All portable electronic devices must remain off and properly stowed during taxi, take-off, and landing.
68. The teaching staff reserves the right to disqualify any device considered unsafe.

**Check-Points****Design Review:**

On **05/07/13** between 9am & 4:30pm we will conduct a design review, one team at a time. Each team should prepare a few sheets of paper showing your proposed designs for both the ROAMER and the POD. You will have 5 minutes to walk us through your ideas. The focus should be on system level concepts, **not detailed hardware or software**. We will spend the balance of the time-slot giving feedback and asking questions. You will present these in 550-122, 550-200 or Ed's front porch (depending on the time-slot chosen).

**First Draft of Communications Standard:**

Due by 5:00 pm on **05/08/13**. Ed will meet with the communications committee on the evening of **05/09/13** to provide feedback on the specification.

**Communications Standard:**

Due by 5:00 pm on **05/10/13**. This is the working draft of the communications standard.

**First Check-Point:**

On **05/14/13**, you must demonstrate

- 1) The ability of the ROAMER to receive and correctly decode and respond to commands from the POD (simulated inputs are acceptable at this time).
- 2) That your ROAMER platform has been built, and is capable of bearing the approximate weight of all the necessary components it will carry when complete. It is encouraged, but not required, to demonstrate working propulsion and steering subsystems.
- 3) The ability to communicate with the SPECIAL and report the SPECIAL status.

The final working version of the communications standard is due. No further changes are allowed to the standard. This protocol will be evaluated with respect to its completeness and suitability for the proposed system. **Note:** this is a functional evaluation only. The focus should be on demonstrating **functional** hardware and software.

**Second Check-Point:**

On **05/21/13**, you must demonstrate the ability to communicate all required functionality between your ROAMER and POD. This will include commands from the POD to the ROAMER and all status messages from the ROAMER to the POD.

**Project Preview:**

At the Project Preview on **05/23/13**, each team must demonstrate (in addition to the 1<sup>st</sup> & 2<sup>nd</sup> check-points' functionality)

- 1) The ability to successfully send and execute the drive, steering and gripping commands (including the actuation of all electromechanical outputs) from an operator of the POD to the ROAMER.

**Grading Session:**

During the Grading Session on **05/29/13**, each team will be required to demonstrate the ability to successfully participate in a mission. This will include

- 1) Establishing communications between your ROAMER and POD, between your ROAMER and the POD from another team and between your POD and a ROAMER from another team.
- 2) Demonstrating the ability to control and display to the POD operator the status of the SPECIAL.
- 3) Navigating a ROAMER from the initial position and successfully opening a hatch and dragging a survival capsule out from behind the hatch.

**Public Presentation:**

This will take place on **05/30/13** starting at 4:00 pm in Thornton 120. At this event, members of the public will be allowed to act as operators of the PODs.

**Report:**

Draft due on **06/3/13** by 4:00 pm. The final version (with revisions incorporated) is due by 5:00 pm on **06/07/13**.

**Evaluation****Performance Testing Procedures:**

One or more of the team members will demonstrate the ROAMER and POD during the first & second check points and project preview. Members of the teaching team will operate the ROAMER and POD during the grading session.

**Grading Criteria:**

- Concept (15%)** This will be based on the technical merit of the design and coding for the machine. Included in this grade will be evaluation of the appropriateness of the solution, as well as innovative hardware, software and use of physical principles in the solution.
- Implementation (15%)** This will be based on the prototype displayed at the evaluation session. Included in this grade will be evaluation of the physical appearance of the prototype and quality of construction. We will not presume to judge true aesthetics, but will concentrate on craftsmanship and finished appearance.
- First Check Point (10%)** Based on the results of the performance demonstrated on 05/14/13.
- Second Check Point (10%)** Based on the results of the performance demonstrated on 05/21/13.
- Preliminary Performance (10%)** Based on the results of the performance demonstrated during the Project Preview.
- Performance (15%)** Based on the results of the performance testing during the Grading Session.
- Report (10%)** This will be based on an evaluation of the report. It will be judged on clarity of explanations, completeness and appropriateness of the documentation.
- Report Review (5%)** These points will be awarded based on the thoroughness of your review of your partner team's report. Read the explanations, do they make sense? Review the circuits, do they look like they should work?

- Log Book (5%)** This will be evaluated by the evidence of consistent maintenance as well as the quality and relevance of the material in the log book.
  - Housekeeping (5%)** Based on the timely return of SPDL components, cleanliness of group workstations as well as the overall cleanliness of the lab. No grades will be recorded for teams who have not returned all loaned materials.
-

## Gems of Wisdom from Prior Generations

- Get the radio working with the 'E128 first.
- Do not continue working until the wee hours of the morning unless you absolutely have to because errors propagate when tired. A fresh look at things in the morning will save you a lot of pain at night. Sleep is not a crutch, it is a necessity.
- Put some time into your first prototype. You might be surprised how many things you throw together for testing purposes make it into your final project.
- Label or color-code your connectors so that it's easy to plug them into the right place. Connectors that can only be hooked up one way (such as Molex) prevent undesirable incidents like reversing the voltage and ground connections and frying components in the process.
- When building networks, add nodes one at a time to better track down "bad nodes".
- Debugging LEDs are useful for getting feedback on the operational state of PICs.
- A "power central" board is a good thing to have, particularly if you're dealing with multiple supply voltages. This makes the circuitry cleaner, and can save you from supplying your PIC with 37 volts.
- Think twice before planning to provide PWM for motors with PICs (at least the one with 20 pins). You will need to take care of output comparators and timers also.
- You will need to leave some pins on PICs (especially those with only 20 pins) open for debugging.
- Don't hesitate to add another PIC and SPI communication. It's really easy.
- Using shift registers for debugging can also be a helpful trick to obtain more information, but it is not good for timing issues.
- Try working during the day (seriously!). Debugging is way easier with a clear head.
- The radio boards runs on 3.3V not 5V. Design your circuits accordingly and be ready to convert between the two voltages.
- Jameco is NOT OPEN on weekends. Don't postpone your trip until Saturday – you will be sorely disappointed.
- Don't be dead set on a theme at the beginning of the project. Let the project theme develop as you move through the project. You'll be surprised how many great ideas pop up as you go along.
- Hot glue down all soldered wire connections. You'll lose a lot of time tracking down an error that may end up being a loose/broken wire.
- Write all functions as non-blocking code – no matter where they fit into the flow of the program.
- Just because two points on a circuit look like ground when probed doesn't mean they are connected.
- Test circuit as it will be implemented in final form, as well as fully integrated.
- Test in environment in which hardware will be used (radios outside, with appropriate distance in between).
- Testing our radio pair in the presence of other active radio pairs revealed problems that didn't exist when we test alone.
- It's easy to make a design with bad ergonomics which make it impossible for the user to perform the task. Prototype/try out the user scenario yourself as early as possible.
- Keep circuit diagrams up to date as you make them.
- Use lab notebook so that all information is at one place and teammates can have easy access to it.
- Take a lot of pictures as you go.
- Remember to HAVE FUN.
- If you are having intermittent problems (e.g. it works only some of the time) check your connections – especially those connecting your various circuits to a common ground.
- Modularize as much as possible – test all of the components separately before integrating
- Build and test all of your circuits and sensors on breadboard before you make them hard mounted on perfboard.
- When moving your circuits from breadboard to perfboard, rather than dismantling your breadboards, leave your working breadboards intact and buy new components and build entirely new circuits on the perfboard. That way, if something goes wrong once everything is built, you will always have a backup copy of your circuits on the breadboard that you know worked before you integrated everything.
- Isolate your circuits onto individual perf-boards (rather than having a giant perf-board with all of your circuits). Makes it much easier to take them out to debug them.
- A nice pair of wire snips (flush cutters) and wire strippers makes wirewrapping and circuit building in general much easier.
- Do your circuit calculations to make sure you have enough/not too much voltage/current/power
- Have plenty of spare parts ready to go in case something blows at the last minute
- Always take the time to test on the actual competition field at the actual competition location
- Make sure at least two people of the group understand or at least have an idea of each component – mechanical, electrical, software. Doesn't have to be the same two people, but it insures that if someone's missing, that the group isn't stuck.
- Be friendly with other teams – you never know when you're going to need help.
- Test your wireless communication outside and at range!
- Test your components for interoperability with everyone else's before game day.
- Source an off-the-shelf housing/controller and gut it. That way you can focus on the electronics and not the mechanical design.
- If you can avoid having to spend time building something by buying an equivalent part, do it!
- Remember your banksel commands and save yourself hours of debugging.
- Make things accessible (i.e. batteries, boards, DIP sockets, etc.) so you don't have to unscrew things when you need to test, power cycle, or reset things.
- Learn to use assembler macros. They clean up your code visually.
- Don't use macros when you can use a function-type subroutine!



- Size does matter. The bigger or larger the motions involved in your controller, the more entertaining it will be to watch.
- Do not bury your wireless antenna in a box. Try to keep it out in the open.
- Buy a good pair of wire strippers, preferably ones that can strip 30AWG wire. Your fingers will thank you.
- Try to avoid having to scramble together parts or code for a check-off. This means keeping on top of the project schedule. If you don't, you will end up throwing away a lot of hours on setups that will not make it to your final design.
- PICs are notoriously difficult to debug. Either source an MPLAB ICD2 (or equivalent), or finish your circuitry early, before writing the bulk of your code. You will find that changing even small things in assembler can cost you hours.
- If a change causes things not to work the first thing you should check is if the code is in the correct bank. It is always a good idea to use a bank select command at the start of every routine rather than assume you'll know where it is.
- Build and test the code in small manageable pieces. If a lot of changes are made at once and the new program doesn't work, it is very hard to isolate the problem without a lot of work.
- Use the debugger. Running routines through the debugger to see what will happen will save lots of time and effort. Getting a routine to work in the debugger usually allows you to assume problems that come up in actual testing are hardware rather than software related.
- When you're tired and everything starts to fail don't forget to check the batteries.
- If you're tired and everything starts to fail and it's not the battery consider going home and looking at it again the next morning rather than changing a lot of code. Often it is some small little change you overlooked and are too tired to notice.
- Make sure all data tables are in the correct location.
- Make use of calls and macros whenever possible to keep the code clean. This also makes repetitive actions easier to code and change.
- Make use of #defines for labeling pins and value as much as possible. This makes it very easy to see what pins are connected to what and allows for the easiest changes. Rather than searching for a specific port and pin throughout the code you only have to change one #define value.
- Don't believe anyone that tells you that the 218C project is less time consuming than the 218b project. It's not.
- Pick your battles early. Learning to program the PIC's and the Zigbees is a lot of work on its own. Trying to add other challenges can be tough.
- Move to solder boards or wire wrap boards as soon as you can. If you are developing simple hardware that you understand well, don't be afraid to solder it on a board. Troubleshooting bad connections on a breadboard is a waste of your time.
- Allocate your pins and subsystems early. A spreadsheet that shows all of your pins is very handy.
- Practice on the course as soon as possible to test your operable range.
- PICs are apparently not designed to be inserted backwards. We recommend against doing this.
- Don't spend more than a few hours debugging SPI code before debugging all of the related hardware.
- Start by making a schedule for the project and include any outside events like vacations, graduations, etc. to avoid surprises later on.
- Black objects left in the sun tend to melt any hot glue that is exposed. This is detrimental to the project's structural integrity. It is therefore wise to a) avoid hotglue or more realistically, b) avoid leaving black hotglued objects in the full sun for extended periods of time.
- Although checkpoints are important, the key is to continue working on the final product, so at all times try to write code/build hardware that you will be able to use in the final product. Try to minimize writing special "check-off code" and building "check-off hardware" that you won't use later.
- Thinking very carefully about your electrical design/layout will save you lots of soldering time. By designing carefully, you'll optimize locations of every components, and you'll end up making a lot less solder joints/connectors/electrical boards, fewer corrections.
- Use Debugging Leds if using PICs. Reserve a few outputs so you can toggle the bits and see if you get into loops or states. This was really helpful when we were trying to figure out what was wrong with our code. Also, since we already used the SSP and Asynchronous communications outputs, we could not use printf's to the terminal.
- Check your #defines and labels. With PIC programming, you tend to have a lot of GOTOs and CALLs which means you need a lot of labels. Try to have a good system for labeling things and creating your constants and variables. We used CAP\_UNDERSCORE for # defines and FirstCapitalLetter with no spaces for variables. Where we went wrong was creating "FORWARD" and "FORWARD\_CMD" which we misinterpreted and messed us up for a long time.
- Talk to other people about the communication protocols and how they implement their code. It's hard to figure out the datasheets by yourself with no help from anyone.
- Debug code extensively prior to integration with other software/hardware elements.
- Utilize 7-segment display or LCD display for real-time debugging.
- Modularize mechanical systems so that simpler parts can be made earlier and used from the beginning of development.
- Develop a clear understanding of the communications protocol from the beginning of code development.
- Communicate well within your team so that some tasks are not overlooked, while others are duplicated.
- Learn some of the common problems with writing PIC code (such as dividing your long code into small sections using the "Maincode code" command).
- Bathe as frequently as possible; encourage others to do so as well.

**Purpose:**

The primary purpose of the communications link to the Solar Powered Environmental Camera Interface with Assistive Lamp (SPECIAL) is to provide information about the state of charge on the SPECIAL batteries and to allow for the control of the assistive lamps on the SPECIAL.


**Interface Connection**

**Connector:**

The connector of the SPECIAL is a 5-pin keyed Molex connector.

**Pinout:**

Pin	Name/Function
1	SDI / Serial Data Into the SPECIAL
2	SDO / Serial Data Out of the SPECIAL
3	SCK / Serial Clock
4	SS / active low select line for the SPECIAL
5	GND / Ground reference for the SPECIAL



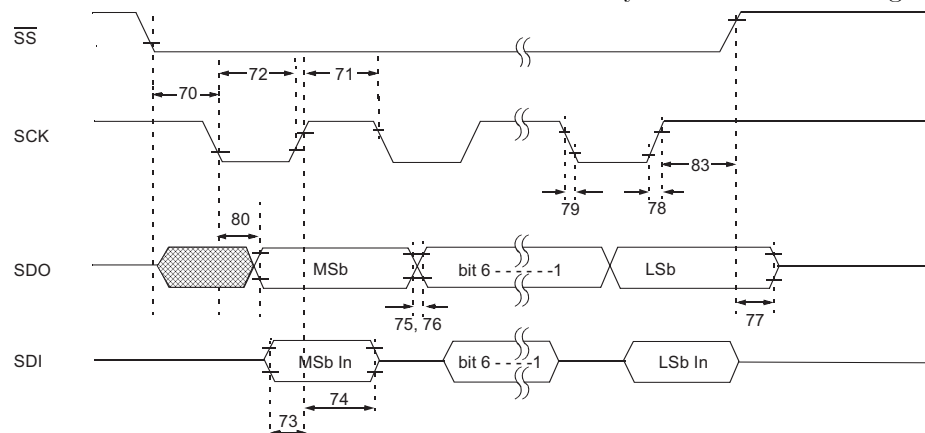
**Electrical Specifications**

Parameter	Min.	Max	Units
$V_{iH}$	$V_{dd} * 0.65$		V
$V_{oH}$	$V_{dd} - 0.4$		V
$V_{iL}$		$V_{dd} * 0.35$	V
$V_{oL}$		0.4	V
$I_{iH}, I_{iL}$		$\pm 1$	$\mu A$
$I_{oH}$	-20		$\mu A$
$I_{oL}$	20		$\mu A$
All Specifications at $V_{dd} = 5V$			

**Byte Transfer Specification**

The SPECIAL uses a synchronous serial signaling method to transfer data into and out of the SPECIAL. The signaling method is compatible with SPI communications, with the SPECIAL operating as a slave device on an SPI network. The  $\overline{SS}$  line must be lowered (asserted) to begin a 4-byte (32 bit) transfer and raised at the completion of the 4-byte transfer. The  $\overline{SS}$  line must remain de-asserted for a minimum of 2ms between transfers. The SDO line represents the serial data out of the SPECIAL, while the SDI line represents serial data into the SPECIAL.

The relationships between the four lines involved in the transfer of a byte are shown in the figure & table below:



Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
70*	TssL2scH, TssL2scL	$\overline{SS}\downarrow$ to SCK $\downarrow$ or SCK $\uparrow$ input	Tcy <sup>a</sup>	—	—	ns	
71*	Tsch	SCK input high time (Slave mode)	Tcy + 20	—	—	ns	
72*	TscL	SCK input low time (Slave mode)	Tcy + 20	—	—	ns	
73*	TdIV2scH, TdIV2scL	Setup time of SDI data input to SCK edge	100	—	—	ns	
74*	Tsch2dIL, TscL2dIL	Hold time of SDI data input to SCK edge	100	—	—	ns	
75*	TdoR	SDO data output rise time	3.0-5.5V	—	10	25	ns
			2.0-5.5V	—	25	50	ns
76*	TdoF	SDO data output fall time	—	10	25	ns	
77*	TssH2doZ	$\overline{SS}\uparrow$ to SDO output high-impedance	10	—	50	ns	
78*	Tscr	SCK output rise time (Master mode)	3.0-5.5V	—	10	25	ns
			2.0-5.5V	—	25	50	ns
79*	Tscf	SCK output fall time (Master mode)	—	10	25	ns	
80*	Tsch2doV, TscL2doV	SDO data output valid after SCK edge	3.0-5.5V	—	—	50	ns
			2.0-5.5V	—	—	145	ns
83*	Tsch2ssh, TscL2ssh	$\overline{SS}\uparrow$ after SCK edge	1.5Tcy + 40	—	—	ns	

\* These parameters are characterized but not tested. <sup>a</sup>Tcy = 33 $\mu$ S

## Byte Level Protocol Specification

### Common Byte Format:

Exchanges between the SPECIAL and your BAT take place with four successive bytes being exchanged. The first byte from the BAT to the SPECIAL is the actual command. The value returned from the SPECIAL during this transfer will be 0x00, but has no meaning. The values sent to the SPECIAL as the second through fourth bytes of the sequence should always be 0x00. The meanings of the values returned by the second through fourth byte transfers will be the results from the command byte.

### BAT to SPECIAL Bytes:

The meaningful values for the command bytes from the BAT to the SPECIAL are shown in the following table:

Command	Meaning
0x3F	Return the state of charge on the SPECIAL batteries.
0xF0 – 0xF7	Activate/De-activate the assistive lamps. The 3 LSBs of this command indicate the desired state for the 3 assistive lamps (Bit0 =right, Bit1 = center, Bit2 = left)

### SPECIAL to BAT Bytes:

The values and meanings of the response bytes returned by the SPECIAL are shown in the following table:

Command	Response Bytes	Description of meaning
0x3F	0xFF, 0xBB,0xCC	BB = state of charge on the SPECIAL batteries (0-255). CC = Charge/Discharge status of the batteries ( 00 = discharging, 0xFE = charging)
0xF0 – 0xF7	0xFF, 00,LS	LS = Lamp status: bit 0 = right lamp, bit 1 = center lamp, bit 2 = left lamp.

### Query the State of the Batteries:

To query the battery charge state, send a byte of 0x3F to the SPECIAL followed by 3 bytes of 0x00. The SPECIAL will process the query and during the three 0x00 bytes of the exchange will return 0xFF, followed by a byte indicating the current charge state, followed by a byte indicating the charging status.

### Command the State of the Assistive Lamps:

To activate/deactivate the state a particular assistive lamp, form the necessary command byte value by bit-wise ORing 0xF0 with the requested state of the three lamps in the 3 LSBs of an 8-bit value. Transmit the resulting

command byte, followed by three bytes of 0x00. The values returned from the SPECIAL during those three bytes of 0x00 will be, in order, 0xFF, 0x00, the state of the three assistive lamps in the three LSBs of this final byte.

### Power on and reset behavior:

Initially, after power on or a reset, the SPECIAL will return 0xFF from any query until such time as the SPECIAL is internally initialized.

### Command Timing:

The interval between two successive transfers from BAT to SPECIAL should be at least 2ms. The  $\overline{SS}$  line must remain high for a minimum of 2ms between successive transfers.

### Invalid Command Bytes:

If the SPECIAL receives a command byte not listed in the table, it will respond to the invalid command byte by queuing a series of 0xFF bytes to be returned to the robot.

### Physical Specifications:

