ADE Component Report

Reducing Food Insecurity by Creating a Low Cost Climate Control System for Small Scale Farmers in Massachusetts

Annie Buchanan, Ben Chapman, Rolando Chinchilla, Ian Daniher,
Dante Santos, Colby Sato, Neal Singer, Kelsey Shilperoort

May 10, 2013

Contents

1 Executive Summary

Massachusetts, USA, May 8th, 2014
Dante Santos

ADE Massachusetts goal is to reduce food insecurity by helping local farms produce more food.

Food insecurity is defined by the World Health Organization as lack of access to sufficient, safe, nutritious food. There are three main causes for food insecurity:
1. Food Availability: for example, a person might live in a food desert where the only local sources of food are convenience stores and fast food restaurants.
2. Food Access: for example, the cost of available nutritious food might be too high.
3. Food Use: for example, knowledge about nutrition, water, and sanitation might be limited.

Food insecurity is a problem that is faced by the people around us every day. As of 2010, over 700,000 people in Massachusetts face hunger. 20% of Boston residents struggle with food insecurity.

Federal and City programs like SNAP and Boston Bounty Bucks attempt to increase food access by subsidizing the cost of nutritious food. Another approach is to increase the availability of food, by encouraging, supporting, and starting local farms, especially farms with social missions. Our partner is one such farm.

Figure 1.1: Serving Ourselves Farm at the beginning of Fall 2013

We have partnered with farm manager Catalina Lopez-Ospina, at Serving Ourselves Farm (SOS), a four-acre organic farm on Long Island in Boston Harbor. SOS directly serves the food insecure people of Massachusetts by working with a homeless shelter and a soup kitchen on the island. The farm is staffed by people from the shelter, its produce feeds people at the kitchen.

Massachusetts has a harsh climate, with extreme and unpredictable temperature fluctuations, and a short growing season. (Fig.1.2) Farmers try to extend the season by starting seeds inside, and letting them grow into small plants before exposing them to the harsh outdoors. This process is called propagation.

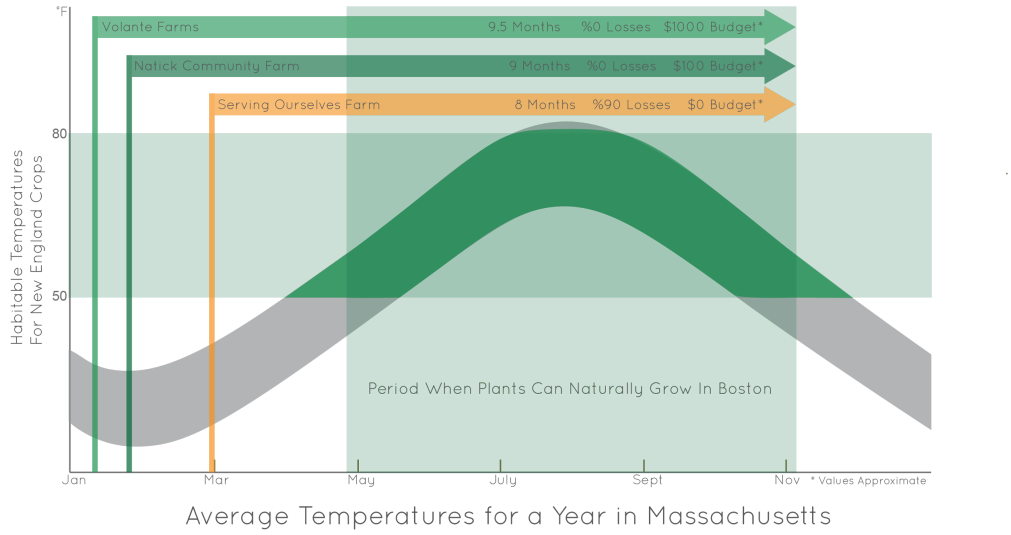


Figure 1.2: A graph portraying how increased budget can extend the growing season of a farm. The grey represents the average temperature in Boston throughout a year, the horizontal green band represents the temperatures at which the most popular crops will sprout from seed, and the vertical green band is the period of the year when the average temperature is within this range. The arrows represent the growing seasons of three farms in Massachusetts.

Some farms can start propagating earlier than others. Farms with bigger budgets can afford greenhouses and other technologies that allow them to extend their seasons further and more reliably. Serving Ourselves has limited available funds, and very little season extension.

Last year, Serving Ourselves lost 90% of its seedlings to a mold caused by poor humidity and temperature control.

Our aim is to decrease seedling death and ease the burden of propagation on local farmers, thereby increasing the amount of food grown by local farms, making more healthy food available to local food insecure people. To do this, we are developing an extremely affordable, wireless, cloud-connected climate control system which we have named SproutWave.

Our project is moving out of the opportunity identification section of the ADE pipeline into development. This semester we have concentrated on user research, technical development, and our plan for this venture post-ADE. This report will discuss the progress of and plans for the six components around which ADE is built, Customer Needs and Market Segmentation, Comparables and Prior Art, Product/Service System, Value Proposition, Business Model, and Social Benefit Analysis.

1.1 Customer Needs

A large portion of our work this semester revolved around research about our users, their stories, needs, and desires. We used frameworks from User Oriented Collaborative Design (a sophomore-level design class at Olin) to better understand our users. These frameworks included personas and a difficulties/needs/requirements table as well as a few more that will not be discussed in this paper such as people portraits and a blue-sky chart.

There were three major assumptions that were tested by our customer needs work this semester: our product aligns with customers values, increased yield and decreased labor is valuable for our customers, and farmers would use a mobile web-interface. In researching this first assumption, we gained a more nuanced understanding of our users while confirming our hopes that we were moving in a productive direction. While speaking to farmers about our second assumption, we learned that farmers would greatly value the added flexibility in their time that our system would provide, rather than doing less work because of it. Finally, we were pleasantly surprised to learn that over 50% of farmers use smartphones.

We currently have a strong technology and user research. The next goal is to have a cohesive product with an appropriately targeted user-experience. This requires integrating our user research with our technical work, drafting web/smartphone interfaces, and testing them with a variety of customer groups.

1.2 Market Segmentation

In order to design for our target users, we need to define exactly who those target users are. A clear and concise notion of how large these markets are and how to reach them is also vital information for developing a sustainable business model. Market segmentation is the process of identifying and determining the size of a product or ventures target customer base. Our understanding of who our target customer has evolved significantly this semester. We moved from designing for small Massachusetts farms with a social mission to addressing the needs of a few very specific market segments. We described five different market segments, three core segments and two supplementary segments. The core market segments affect the food insecure population in Massachusetts, and the supplementary segments provide enough potential customers to guarantee that the venture could be sustainable.

There were five assumptions that we tested this semester: customers will measurably impact food insecure people of Massachusetts, existing products do not meet the needs and situations of our customers, our product is affordable enough for our customers, our product is applicable to multiple market segments, and A cross-subsidy model could be used to serve multiple markets at affordable prices. These were tested by internet research, phone calls, farm visits, and farmers markets visits. We were disappointed in the number of farms that were actively serving the food insecure, but we found a number of ways to track how much they do serve the food insecure (food donations, SNAP use, etc.), and determined that our target customers do in fact serve the food insecure. The farmers that we spoke to really like the idea of having an automated climate controller, but do not buy the existing products on the market because they are unaffordable and don't have the flexibility and remote-operability that make them useful on small farms. We also found that our product would also be well-received outside of our target market segment, and that it would be extremely valuable to consider a cross-subsidy model in the future.

The next steps for this component are to inform the Customer Needs component about the two most relevant customer segments, Beginning Farms and Social Mission Farms, as well as to do a more thorough study of the number of these farms and their demographics.

1.3 Comparables and Prior Art

In order to develop a unique solution that addresses problems better than any previous product, we researched both products being used for our intended purpose in the field currently and comparable products being sold. Previously we had only been able to find larger-scale climate control solutions, so this semester we

looked for more small-scale agricultural systems from which we could learn more.

While our product improves upon many currently available climate control systems, several beneficial characteristics that we either have been planning to implement, or could implement to improve the system, came to light through this research. Our product is already superior because it is capable of controlling a near-infinite number of outlets, can log data and display it remotely, and is affordable to even the smallest of farms. We learned that we should consider also incorporating an alternate power sources such as a battery or solar power, waterproofing and general ruggedization, and a clean user interface.

More research should be done into these in the future and they should be ranked to determine their relative desirability. Additionally, research should be done into comparable business plans and distribution models.

1.4 Product/Service System

Our product is an extremely affordable automatic climate control system. It consists of a set of wireless sensors that take readings about a variety of environmental factors (such as humidity, air temperature, soil temperature, soil moisture, acidity, power use, etc.), a small computer that processes this data, and a set of remote outlet switches wirelessly controlled by the computer. The outlets power the environment control products that the farmer already has (heating mats, propane heaters, watering systems, lights, fans, etc.) so that the farmer doesnt have to turn them on and off his or herself, modulating the environmental factors so that they are optimal for the seedlings.

The current hardware package consists of a receiver, transmitter, computer, remote sensors, and remote outlets, all bought cheaply from third party suppliers. The truly innovative part of the product is the software that ties all of these components together. The sensors and outlets communicate over a globally standard industrial, scientific, and medical band of radio waves, which the computer picks up with a USB radio receiver. The transmitter is still in development. We have developed an incredibly efficient decoding software to communicate with the sensors and outlets, which means that the computer that the software runs on doesnt need to be particularly computationally powerful. There are three main parts to the software system, two built in Rust that comprise the backend that do the communications, and the frontend, which uses Apitronics Hive, a Node application developed by Apitronics, the comparable discussed in the Comparables and Prior Art section. This frontend is currently under development.

Next steps on this component include ruggedizing the system, developing the frontend user interface, and developing the control algorithms.

1.5 Value Proposition

Any venture needs to understand the value that their product or service will provide to various stakeholders, so that they can frame their own actions as well as sell themselves to potential investors and customers. Phrasing this is especially important for a social-good based venture because profit margins, which are easy to calculate and explain, are not the point. Being able to quickly introduce people to the idea of your venture and convey to them its importance and usefulness is vital for a social ventures success. This succinct statement of importance is the value proposition.

This semester, through our user research, we gained a much better understanding of what aspects of our system are most immediately desirable to our users. Many farmers spoke of the stress of propagation, how

intensive the work is, and how easy it is for everything to go wrong in just a few hours. They also spoke to us about the extreme financial constraints that farmers face, and how exciting it was to see this sort of product in an affordable price range. With the insights gleaned from our conversations with farmers, we crafted a new value proposition.

We tested two assumptions and one question to help us form this value proposition: increasing yield during propagation would enable farmers to extend their season, farmers in our market segment would gain financial benefit from a climate control system such that the climate control system would be worth the investment, and what are farmers willing to pay for the system based on the value proposition we communicate? Indoor propagation itself is a form of season extension, allowing plants to start growing before the outdoors is warm enough, but we found that it is unlikely that our system would provide the ability to grow more food, only make the process of growing seedlings easier and more reliable. The system is confirmed to be a financial benefit for the farms, but the exact benefit is hard to quantify. More work should be done on this assumption next year. From our conversations, we have found that how much a farmer is willing to pay depends on the size of his or her farm. A small farm might find the remote monitoring function handy, but might not need help controlling the climate as much as a larger farm with multiple greenhouses.

Next semester, the team should talk to more farmers, gathering information about the financial benefit of the system as well as more narratives about how the system is/might be useful, as those are more illustrative. It would also be beneficial to look into alternate market segments and uses and develop value propositions for them.

1.6 Business Model

In a social venture, a well thought-out and executed business model is vital to make maximum impact while remaining sustainable for as long as possible. Because we have a clear direction for our product/service system, we have begun developing a business model. We, as a team, ideated a variety of potential business models, weighed the pros and cons, and decided on a preferred way to leave the ADE pipeline.

We considered four different models: become part of the product line of an already established business, develop our own independent startup, be part of a government program to subsidize and distribute the CCS, or develop a do-it-yourself instructional kit for farmers to build their own CCS. We decided that developing our own startup was the most likely to be financially sustainable, agile, and high-impact. We discussed branding, and have decided to call the product/venture SproutWave.

Figure 1.3: The SproutWave logo, designed by Brett Rowley, Olin class of 14.

We have done initial calculations to see if SproutWave could be sustainable as an independent startup. Initial calculations indicate that if we start the company while still being part of ADE and receiving ADE

funding, we could survive the larger costs of startup and become a sustainable venture.

We tested three assumptions to verify this:

1. The price of our product is both affordable for our market and feasible for us
2. We have the financial resources to start our business independently
3. We could sell our CCS both through greenhouse equipment retailers and online retailers

Assuming we can purchase our off-the-shelf parts for a total $65 for a unit at quantity 500, and we are selling the product for $150, we could be sustainable. If ADE fronts the startup costs, we only need to sell 13 units to break even and pay them back. In order to afford the next bulk order of 30 units would only need to sell an additional 17 units. We spoke to retailers and found that they are interested in selling our product, as they sell other systems that cost much more but do much the same thing. Additionally CoolBot is a similar system, but for refrigeration, that has been extremely successful with selling online.
Next semester the team should look further into the details of the business plan such as who will do assembly both before and after the project leaves the ADE pipeline, how maintenance is performed, whether there is a service that accompanies the product where a system is customized to a farms needs by someone from within the venture, and developing partnerships with organizations that work with food insecure people.

1.7 Social Benefit Analysis

Social benefit analysis is used to evaluate the impact of a product. Conducting such an analysis involves determining inputs to and outcomes from a system and assigning monetary values to them to provide a common unit so that they can be compared. The outcomes value is adjusted to account for what would have happened in the absence of the product, and then is divided by the total inputs to the system to get a Social Return on Investment ratio (SROI ratio). This ratio represents the additional social value that can be achieved with the product. We have been working on finding ways to quantify the inputs and outcomes, as well as finding all the relevant inputs and outcomes. We have created a framework with initial values, but further work can be done to refine the analysis. More accurate numbers can be found once we have prototypes in the field, and more inputs and outcomes will result from research and co-design.

2 Customer Needs

Massachusetts, USA, May 8th, 2014
Neal Singer

2.1 Introduction

Our project focuses on addressing the issue of food insecurity in Massachusetts. Our theory of change, informed by food policy experts, is that growing more local food will directly benefit the food insecure, hence the target audience for our product is small farmers. The goal of our project is to develop an affordable climate control system (CCS) that will help farmers produce more food by extending their growing season and decreasing both the propagation mortality rate of their seedlings as well as the amount of labor required during propagation. This section of the report focuses on our projects Customer Needs component, addressing who are customers are and identifying their needs. The work on this component last semester focused on the needs of one user–Serving Ourselves (SOS) farm in the Boston Harbor. We have been actively moving forward in the development stage this semester and have accordingly expanded our user research. Our major advances this semester include gaining a depth of insight through user interviews and site visits (condensed into personas), and generating preliminary ideas of how the user experience should be made to meet customer needs. Major pivots this semester including discovering that our product could be made to meet the needs of many types of growers, elaborated on in the Market Segmentation component of this paper.

2.2 Description

The work this semester largely focused on forming personas, which are composites of people we interviewed. These composites allow us to pull the aggregate trends out of our collected interview data, and condense them into clear data points such as needs, values, traits, etc.

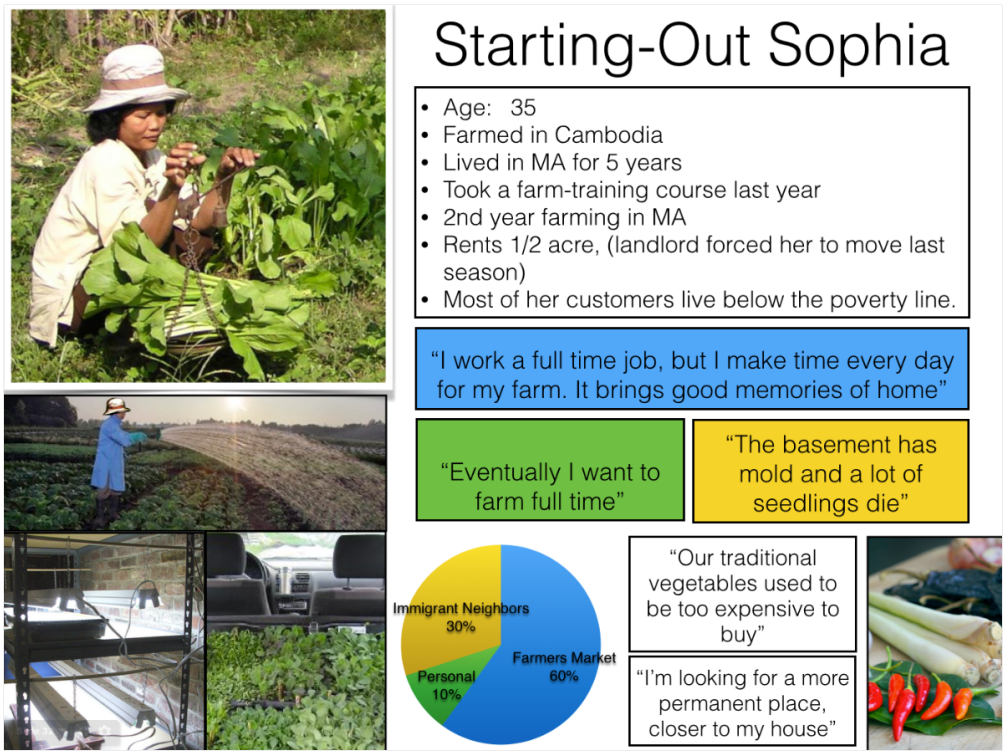


Figure 2.1: Simplified persona for Starting-Out Sophia.

Our first persona is Starting-out Sophia; Fig.2.1 is a much condensed representation of her persona, and the full version of her persona is in Appendix B.

Sophia is a Cambodian immigrant to the U.S. She moved to MA right after she arrived in America, and has been here for five years. She makes her living as an electronics assembly technician for a large defense contractor, and lives in a duplex with her father, husband and two children.

Sophia loves to garden. In Cambodia she grew most of her own food, and farming is in her blood. Growing food now brings back good memories of home and allows her to cook with traditional vegetables she otherwise wouldnt be able to afford; she also sells these vegetables to people in her community to help them out and earn a little money for herself. She says that One day I would like to quit my job and farm full-time.

Sophia grows on an urban half-acre plot several miles from her house. She currently propagates in her basement, and when her seedlings are big enough she drives them across town to plant them in her plot. Propagation causes her a lot of trouble; she constantly deals with mold that kills her seedlings and she is unable to control the temperature of her basement–on some winter days shell wake up to find her seedlings got too cold and froze to death. Transporting her seedlings, too, puts them under a lot of stress as they are rattled and jostled on the bumpy roads, and she wishes she could propagate them on-site.

Based on the difficulties in her farming experience, some of the opportunities that our CCS holds for Sophia include: It would help enable her to become a full-time farmer; propagation can be a large time sink for any farmer, and our system would remove a large portion of that burden from her while at the same time allowing her to scale-up her operation. It is wirelessly controlled and largely autonomous, allowing her to propagate on-site with only a simple greenhouse required. Through temperature control and humidity monitoring, it solves both her temperature issues and her mold problems. Its wirelessness also makes it unobtrusive if she

chooses to continue propagating in her basement.
The above opportunities and Sophias context highlight the following needs:

1. An implicit need for inexpensive: Sophia supports her family, and does not have the means to afford expensive equipment for what is now just a farming hobby.
2. A need for the system to recognize when environmental conditions are going bad, and to act accordingly, controlling environmental actuators
3. A need for wireless communication, both between the base station and sensors, and between the base station and the computer/smart phone that the user controls it from.

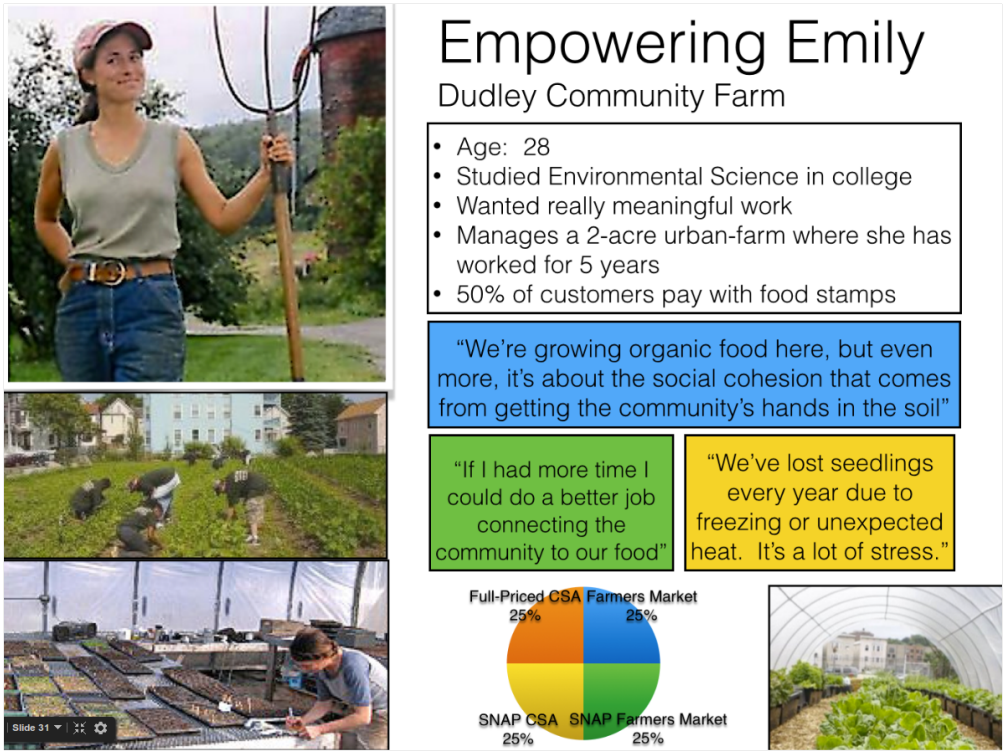


Figure 2.2: Simplified persona for Empowering Emily

Our second persona is Empowering Emily; Fig.2.2 is a much condensed representation of her persona, and the full version of her persona is in Appendix B.

Emily is the manager of Dudley Community Farm, which is on the outskirts of a Boston suburb. She is college-educated, with a degree in environmental science. She wanted to do something more meaningful with her life after she graduated, and chose to go into farming. She has been managing the two-acre Dudley farm for the past five years, and loves the community impact it has, saying: We're growing organic food here, but even more, it's about the social cohesion that comes from getting the community's hands in the soil."
The farm operates a low-income farmers market and CSA, and half of the food grown is sold to persons with food stamps. Emily employs three farmhands to help with the farm work and has a greenhouse for propagation, which was paid for with grant funds. She runs farming education courses and community outreach programs out of the farm, and wishes she and her staff didnt have to spend so much time tending

the greenhouse, having to check on it every hour and roll up the sides if it gets too hot or down if it gets too cold. She feels her time could be better spent: "If I had more time I could do a better job connecting the community to our food."
She has lost seedlings every year, and propagation is a major source of stress for her: she says The greenhouse is like tending an infant on its deathbed. Some mornings the sun comes out earlier than expected and the seedlings roast to death before she is even awake. Sometimes the thought of this even keeps her up at night worrying.
Based on the difficulties in her farming experience, some of the opportunities that our CCS holds for Emily include:

1. It gives her more time to do the community outreach she values, by automating the repetitive, time-consuming tasks associated with propagation in a greenhouse: if it gets too hot, the CCS turns on a fan; too cold, it turns on heating mats; and if the plants need water it turns on a pump

2. It gives her peace of mind: when she first gets our system, she checks the greenhouses status compulsively with her iPhone, but over time comes to trust the system and is eventually able to free her mind of most worries she once had from the greenhouse.

3. It saves her lost time and money: a tray of plants represents an investment of both time and money. Not having this value disappear if the weather report is incorrect is a boon to a non-profit farm that is mostly supported by grants and community efforts.

The above opportunities and Emilys context highlight the following needs:

1. An implicit need for affordability: The Dudley Community Farm is a non-profit, possibly with a board of directors who approve equipment purchases; justifying a several-hundred-dollar or more purchase to them could prove difficult.

2. A need for the system to recognize when environmental conditions are going bad, and to act accordingly, controlling environmental actuators.

3. A need for an option to have the system to clearly inform the farmer of what its doing, so he/she gains trust in it; e.g. having it text you The system detected your greenhouse got too hot. It turned on fans and is now cooling at 10 degrees per hour

4. A need for wireless monitoring, allowing Emily to know that the seedlings are okay.

5. A need for wireless control both between the base station and sensors, and between the base station and the computer/smart phone controlling it; Dudley repurposes the greenhouse during different times of the year, and having to deal with wiring would be a major chore for them.

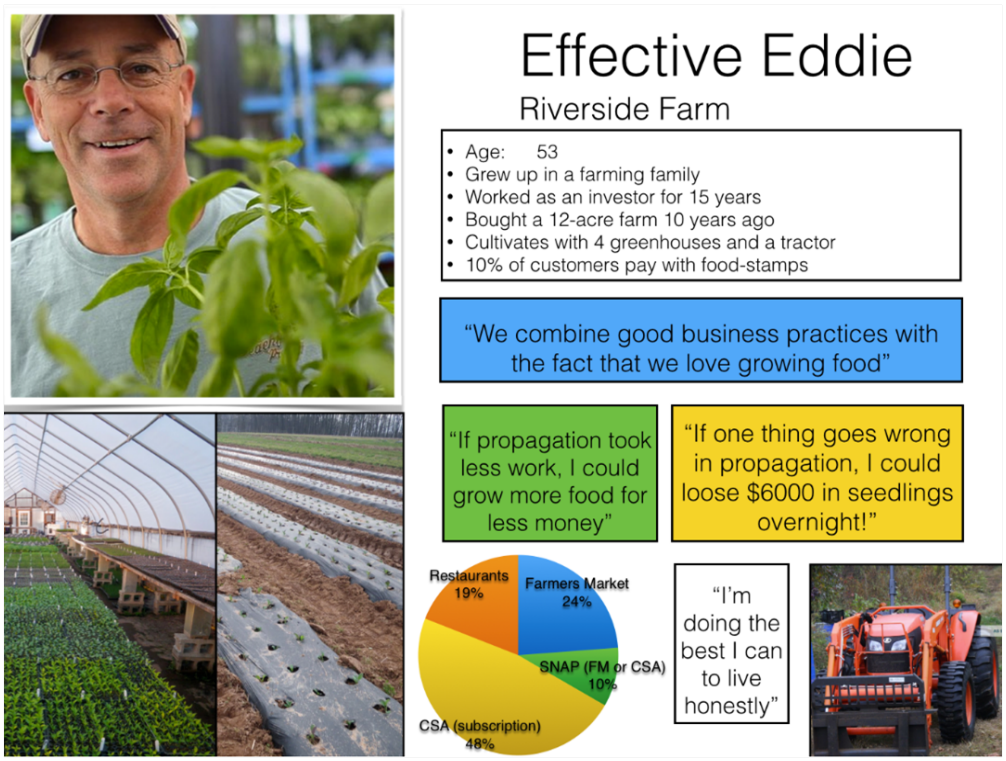


Figure 2.3: Simplified persona for Effective Eddie.

Our third persona is Effective Eddie. Fig.2.2 is a condensed representation of his persona, and the full version of his persona is in Appendix B.

Eddie is the owner and operator of Riverside Farm. Riverside is a twelve-acre, for-profit organic farm on the banks of the Concord river near Lexington. Eddie cultivates with a tractor, four greenhouses and a staff of five workers.

Eddie grew up on a farm, but decided to go to school for business and worked as an investment banker for 15 years. Ten years ago he quit his job, took his savings and bought a rundown farm near home, and he has been growing ever since. He likes farming because it lets him do the best he can to earn an honest living, and says We combine good business practices with our love for growing food. Eddie is a pragmatist who sees the farm as a business, and he couldnt keep running Riverside if it werent a financially successful venture. Eddie wants to grow his business like he grows plants, and is open to buying more greenhouses and equipment to increase his farms success.

Eddie knows what his customers want, and carefully chooses what he grows to suit them. The four greenhouses allow Eddie to have separate microclimates to propagate multiple varieties of plants that require disparate conditions. He says he does propagation right, but it takes a lot of man-hours. The greenhouses also extend his season through the winter. This requires lots of propagation, and Eddie recognizes the risks that entails: If one thing goes wrong in propagation it could cost me $6000. Eddie recognizes that if propagation took less time, he could grow more food for less money.

Eddies years growing up on a farm saw his family go through some difficult times, so he donates the food that he doesnt sell.

Based on the difficulties in his farming experience, some of the opportunities that our CCS holds for

Eddie include:

1. Modularity and expandability: Eddie picks his crops based on the changing tastes of his customers, and this means he has to reconfigure the microclimates in his greenhouses, potentially requiring different sensors or actuators. Eddie also views scaling up his operation as important, and would like a system that is easily expandable

2. Breaking down an impediment to expanding: Eddie recognizes the importance of doing propagation right–if his seedlings die, hes lost irreplaceable time in the season and lots of money in seeds and right now he is impaired in expanding his operation by the effort and labor required in ensuring propagation is done right. Our system automates the majority of the important and time-consuming tasks of propagation, and solves this barrier.

3. Decreasing the cost of business: As mentioned in the point above, doing propagation right takes a lot of resources, and if it were easier the decreased costs would flow through to the consumer.

4. More mental energy: Eddie is a businessman, and gets very stressed about the state of the seedlings which are the lifeblood of his farm. Taking this worry out of the equation allows him toi focus on more important things, such as getting more food to more people.

The above opportunities and Eddies context highlight the following needs:

1. A need to be expandable and modular. The system needs to capable of being broken down and reconfigured to meet different growing climate needs.

2. A need to be wireless. At twelve acres, Riverside farm is large enough to be a pain to wire our system into. Also, as part of the modularity aspect, it would prevent confusion in how everything is hooked up if the system is wireless (you dont want it looking like the back of a stereo/speaker system).

2.3 Requirements

Difficulties	Needs	Requirements
Personal Struggles/ Conflicts		
Plants get too hot/cold/humid and die	System needs to recognize when environmental conditions are going bad and act accordingly	System should have a temperature sensor, recognize when the conditions are trending toward dangerous levels, and react by turning on a fan.
Farmers don't know when conditions are bad unless they're on location	System needs to function independently in the absence of the farmer	System should have automatic control loop
	System needs to alert the farmer if conditions will go bad	System should be able to autonomously detect when conditions are going bad
	Farmer needs to be able to check the current conditions and trends easily	At-a-glance system status and direction, built into interface
Farmers have a lot of stress around ensuring the safety of the seedlings	System needs to be so reliable that the farmer will no longer worry.	Very low failure rate, ability to show the farmer that good decisions are being made.
Farmers spend a big portion of the day tending to indoor growing conditions (opening/closing vents, turning on and off heaters)	System needs to be able to do repetitive tasks at precise times to regulate climate	System should control environmental actuators
Farmers want better control, but can't afford expensive equipment	System needs to be affordable	System should be less than $200.
Farmers want good equipment, but don't want things that will break/ become irrelevant.	System needs to be quality	System should have a 2-year warranty
Farmers must use greenhouse for multiple purposes throughout season	System needs to able to be moved around easily	System can be set up and broken down quickly and easily
Farmers are too busy to learn complicated technologies	The system needs to be straightforward and easy to use	System should have a clear, simple manual.
Implicit Needs		
Farm work stresses equipment/equipment is exposed to the elements	System needs to have a guarantee of some sort	System should have a 2 yr minimum warranty
Farmers may not speak English	Systems needs to not rely on user's understanding of a particular language	System should use symbols as much as possible to communicate

Figure 2.4: Requirements table that sums up the difficulties and needs that we have noted from our work this semester.

2.4 Assumptions and Testing

We identified three major assumptions for the user needs component of our project, as listed in Fig.2.4, and have a fourth assumption identified, that should be explored in the Fall semester. Each assumption, its implication and how we tested it, are as follows. These assumptions affect each of our personas, and directly inform our ideas of the users interactions with, and opinions of, our product.

Assumption 1: Our product aligns with customers values. We initially thought farmers might view our system as threatening, because it would be replacing an aspect of their job with an automated machine.

Test for Assumption 1: We verified that our assumption was true with the $5 test of talking to farmers.

Findings for Assumption 1: We found that farmers would rather spend more time in the field doing work rather than being in a greenhouse watching over their seedlings. We also found that propagating seedlings is (to quote one farmer) like watching over an infant on its deathbed, and they would gladly welcome a system that reduced the stress of this responsibility. We found this to be a powerful insight, and we have attempted to encapsulate it in the personas above.

Assumption 2: Increased yield and decreased labor is valuable for our customers.

Test for Assumption 2: We started out fairly confident in this assumption, but were able to become very confident in it with further $5-test interviews.

Findings for Assumption 2: In the status quo, some farmers have to spend the majority of their day in a greenhouse, making sure it doesnt get too hot/humid/cold. This is a rather simple job, and it can be frustrating for them to have to perform such simple tasks when their skill set could be used much more effectively in the field. The farmers we spoke to unanimously said that they would gladly use our system if it gave them more flexibility in their time.

Assumption 3: Farmers would use a mobile web-interface.

Test for Assumption 3: $5 conversations with farmers validated this assumption.

Findings for Assumption 3: Over 50% of farmers we spoke to had smartphones (brand/OS of smartphones was mixed, it wasnt all iPhones). This assumption decreased in importance to us, because we decided it was important to move towards a more web-based (i.e. from a computer) interface as well as noting the need for a means of monitoring/controlling the system while inside the greenhouse (see product/service system component for more info).

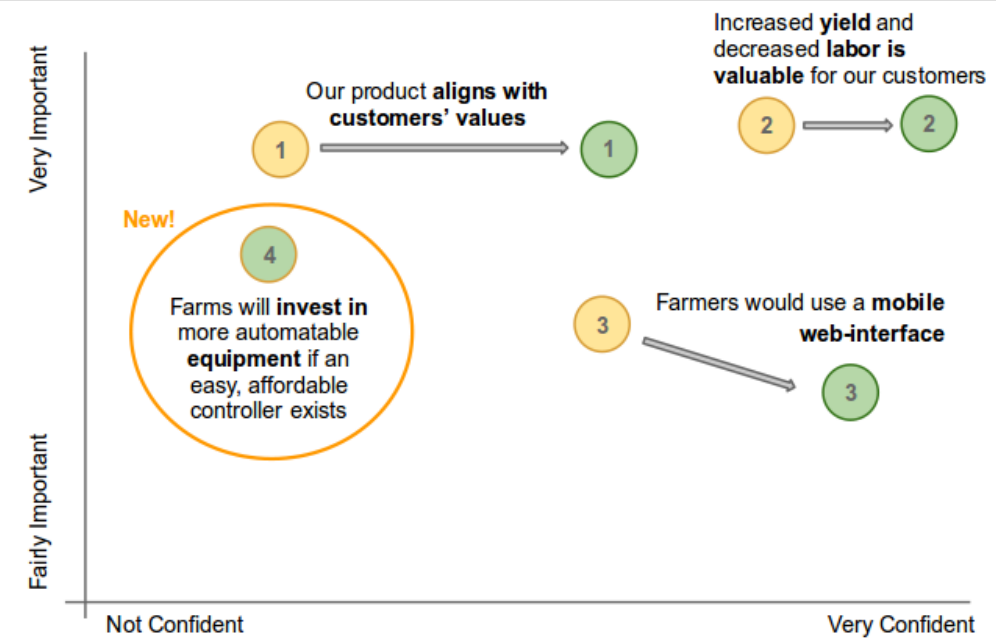


Figure 2.5: The current state of our assumptions as of CPR 2, and their progress since CPR 1.

2.5 Refinement

Since our work this semester focused on getting to know the user-base for our project, we did not face very many major fork-in-the-road decisions. Perhaps the two major decisions we faced were:

1. Deciding how many user visits to do, where to go, and when.
   Our team recognized the importance of doing as many user visits as possible, as quickly as possible, in order to get the greatest amount of information possible. However, you need to balance this with the teams workload and avoid burning out/prioritizing visits over things that may be more pressing. We began the semester by front-loading visits to our pilot farm, farms in the area and farmers markets. The visits, beside those to the pilot farm, mostly took place on weekends, but we found that this was not sustainable for us to do this every weekend. As we acclimated to the semester, we found two visits a week to our pilot farm worked out well, since half our team could make it one day, and the other half could make it the second day. This, combined with an occasional weekend visit to other farms/markets, proved to give us adequate amounts of interaction with farmers and not be too demanding on our schedules.

2. Deciding what to put into our personas.
   The personas we created encapsulate a large amount of the information weve found about our potential users. However, since there is a limited amount of data that can go in a persona, choosing which points were important and relevant was a crucial aspect of developing them. We initially created some people portraits of real farmers we had interviewed, outlining key quotes and demographic information about each farmer. These guided the crafting of our personas, but we obviously couldnt include every piece of data from them. The compromise we found was to group the people portraits into themes (e.g. large-scale farmers, social-mission farmers, beginning farmers), figuring out general demographic information

for each theme (e.g. large-scale farms have acreage from 12 to 90 acres, social-mission farms range from 2 to 12 acres, social-mission farmers tend to be women between 25 and 50, large-scale farms tend to be men between 35 and 68, etc.) and pull out quotes that exhibited extreme emotion. Sometimes, if a sentiment recurred enough, we averaged the quotes to provide a sense of the overall feeling of the farmers.

2.6 Conclusions

This component report presents a lot of information on the personas we created this semester, the needs we identified for them and the requirements that these needs inform. A clear step forward next semester would be to take this data and turn it into actual features for the CCS. These need not be implemented in coded interfaces; a good idea would be to make paper mock-ups/props and go do co-design with the farmers we worked with this semester. Laying it out as a process, I would approach it like the following:
Personas/requirements table
Create features list
Make props/paper mock-ups
Do co-design
Update requirements table
Update features list
New props/mock-ups

As the features become more concrete, make sure to stay in close contact with the technical team, since they will be the ones implementing the features, and the business team, making sure that the features align with how we segment our product.

3 Market Segmentation

Massachusetts, USA, May 8th, 2014
Ben Chapman

3.1 Introduction

Our team is working to reduce food insecurity in Massachusetts and the surrounding region by developing technologies that enable local farms to produce more food for less money. We are developing a product makes the seedling propagation process more effective, less stressful and less time consuming for farmers by electronically monitoring the climate, notifying farmers if things go wrong, and intelligently managing the seedlings environment for optimal growing conditions. This section of the report discusses market segmentation, which is our investigation of who our customers are, how we can group them based on their needs and situations, and estimating how many customers we expect to find in each group.

At the beginning of this semester we defined our target customers vaguely as small Massachusetts farms with a social mission. We now have a much clearer vision of which types of farms we want to sell to, how the different types have somewhat different needs and values, and how many farmers there are of each type in Massachusetts and the United States. We gathered this information from doing many interviews and farm visits, looking through documentaries and articles about farmers, and analyzing agricultural census data. We are using our knowledge of our markets to inform business decisions including pricing, scale, and distribution methods, and especially our value proposition. We also use it to inform technical decisions including interface design, product requirements, and scale.

3.2 Description

3.2.1 Total Market Size

In total, we estimate that our core market size in Massachusetts is 1350 farms, and 390,000 farms in the USA. To give some comparison, the USDA published in 2012 that there are 145,000 farms in the US that sell products directly to consumers. We would expect our market size to be very close to this number, so our estimate is probably high.

We used data from the USDA Census of Agriculture to estimate our numbers with help from other sources and from our own experience. Fig.3.1 shows some of the more useful statistics from the census report. The census defines a farm as any place from which $1,000 or more of agricultural products were produced and sold, or normally would have been sold, during the census year." There are 7800 farms in Massachusetts, but only 20% of those are growing fruits and vegetables. Our target markets all fall within that 20%. (The

other 80% of farms are mostly farms that raise livestock and poultry(45%), farms that grow hay (15%), and nurseries that grow flowers or non-food plants (13%)). In the future it would be excellent to get data that is granular enough that we can just see statistics about the types of farms and farmers that we are trying to serve.

If we assume local fruit and vegetable farmers are characteristic of all American farmers, the data tells us some about farmers characteristics. The average farmer age is 58, so they may have difficulty adjusting to an electronics-based technology and workstyle. Half of all farmers have other primary occupations, they work other jobs, so they would appreciate the ability to monitor and control growing conditions remotely. Six percent of farms are run by people under 35, so there is a small but notable group of young farmers that may serve as early adopters for our system. Were also excited about the 10% of farms that are less than 5 years old and the 10% of farms that are nonprofits. These correspond to our beginning farms segment and social farms segment.

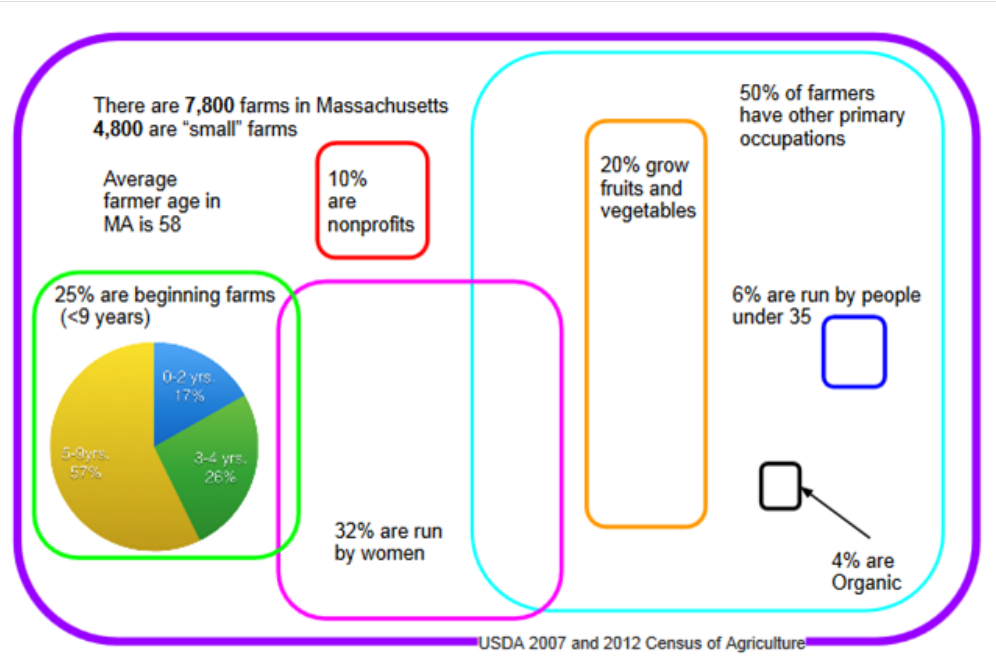


Figure 3.1: Categories of Farmers in Massachusetts.

3.2.2 Core Market Segments

We have identified three core market segments. These are market segments that have the potential to increase access to fresh, nutritious food for local food-insecure people, by using our system and have seedling-propagation needs that our system can fulfill. The three segments all consist of local farms that sell their fruits and vegetables to consumers within the same geographic area (usually within a one-hour drive). Usually they sell directly to the consumer through farmers markets and subscription-like CSA shares. They do not include the giant factory farms that produce the majority of the grains, meat, and vegetables that end up in American grocery stores and restaurant chains.

Figure 3.2: Our three core market segments; (a) For-profit farms; (b) Non-profit farms; (c) Beginning farms.

The first segment is comprised of established, for-profit farms (Fig.3.2 a) that grow fruits and vegetables and sell locally. These farms typically have ten to two-hundred acres, and a commercial mission (that is, they raise food to make money). They usually own their own land or have long-term leases. These farms typically have several farm workers, a tractor, and multiple greenhouses. Some grow food all year long. Only about ten percent of their customers are food insecure. They tend to make decisions based on business-logic, but sometimes they are happy choosing to make less money and doing more good for their community. While they may not explicitly aim to help the food insecure, these farmers are important because contribute to the overall food supply, and if their propagation efforts took less work, they could grow more food for less money, positively affecting, the food insecure, those at risk of becoming food insecure, and those who purchase food to feed the food insecure (e.g. food banks).

In Massachusetts, there are about 1500 farms that grow fruits and vegetables, but not all of them sell locally. We estimate that about of those farms fit in our established for-profit farm segment giving us a total of about 500 farms. Nationwide, the number of these farms is about 200,000 based on those same assumptions.

Our second segment is comprised of nonprofit farms with a social mission (Fig.3.2 b). These farms tend to be smaller, from two to seven acres in size. Some of them are urban farms. Often their land is owned by a town or nonprofit organization. The farmers tend to be college educated and are drawn to farming because they seek really meaningful, impactful work. The farms may have some full-time farm workers, but almost always work with some volunteer labor. These farmers often enjoy doing education courses and community outreach events related to farming. Our system assists them by helping them to propagate seedlings more effectively, allowing them to provide more fresh food to members of their community and to the hungry. Our system also frees up up the farmer from watching over the greenhouse for large portions of the day, allowing them to do more of the community outreach that they enjoy. The number of these farms is difficult to estimate, but we do know that 10% of farms in Massachusetts are not for-profits. We can estimate that

about half of these farms are within our market segment. Totalling about 375 farms in Massachusetts. Scaling this up to the national scale, we estimate that there are about 100,000 of these farms in the US.

Our final core segment is comprised of beginning farms (Fig.3.2 c). These are small farms of two acres or less, that have started in the last few years. Many of these farms belong to immigrants, often living close to the poverty line. They typically rent their land, on short-term leases, so they could get kicked off with very little notice from the landowner. These farmers often like to grow culturally relevant foods, especially at affordable prices. These are typically working-class folks, and our system would benefit them by watching over their plants while they are out at their jobs, and would give them back precious time in their day they would otherwise spend doing chores such as watering and monitoring climate conditions. In Massachusetts, 11% of the farms have been around less than 5 years. We assume roughly half of those farms are in our market segment, resulting with an estimate of 470 farmers. This is very similar to the number of Massachusetts farms are run by people under 35. Nationwide, we estimate that there are about 90,000 of these farms, based on those same assumptions.

3.2.3 Supplementary Market Segments

We also have decided to investigate two supplementary market segments. These are market segments that have propagation needs that could easily be met by our core technology and can afford to pay for a higher margin product. Expanding to supplementary markets would also increase our economies of scale. If we choose to design for these markets it will make product development and branding more difficult. To make the most of these markets, well need multiple brand images, multiple interface designs, and perhaps multiple product designs and service offerings.

Figure 3.3: Our two supplementary market segments; (a) Home gardeners; (b) Medicinal herb growers.

Our first supplementary market segment is home gardening enthusiasts (Fig.3.3 a). These are avid gardening hobbyists. They are knowledgeable about best planting practices, disease avoidance, and plant varieties. They get a lot of pleasure out of gardening, especially from producing their own fresh fruits and vegetables. There are about equal numbers of men and women, equal numbers in suburban and rural locations, over 70% are married, and just over 50% have been gardening for over 10 years. On average, they spend $700 per year on gardening supplies and equipment, over twice the national average for spending in those areas. Home gardeners are well studied in the US by a group called the National Gardening Association. They estimate that 9% of Americans fit in the gardening-enthusiast Category, totaling roughly 28 million gardening enthusiasts. This market is expanding by about 19% per year as more and more people become dedicated gardeners. Figure incredible shows more information on the characteristics of American food-gardeners.

We have not yet interviewed people in this market segment, but weve observed that these consumers like to buy gardening equipment that makes gardening more fun, more productive, and feel less like work (Fig.3.4). We think that with SproutWave, these customers would love the ability to feel connected to their plants from wherever they are. They would appreciate having higher seedling yields, and reducing the number of monotonous tasks involved with growing. Enthusiasts may value having the latest and greatest in

gardening technology. Most of the customers in this market segment are not food insecure, but we predict that they would be willing to pay a higher price for a version of our system designed and marketed specifically for them. This high-margin market could help us sustain our business while selling systems at very low margins to local farmers. We are also exploring the idea of a product offering directed toward food insecure people who are interested in growing their own food for the first time. We could lower the barrier to entry for people helping themselves out of food insecurity.

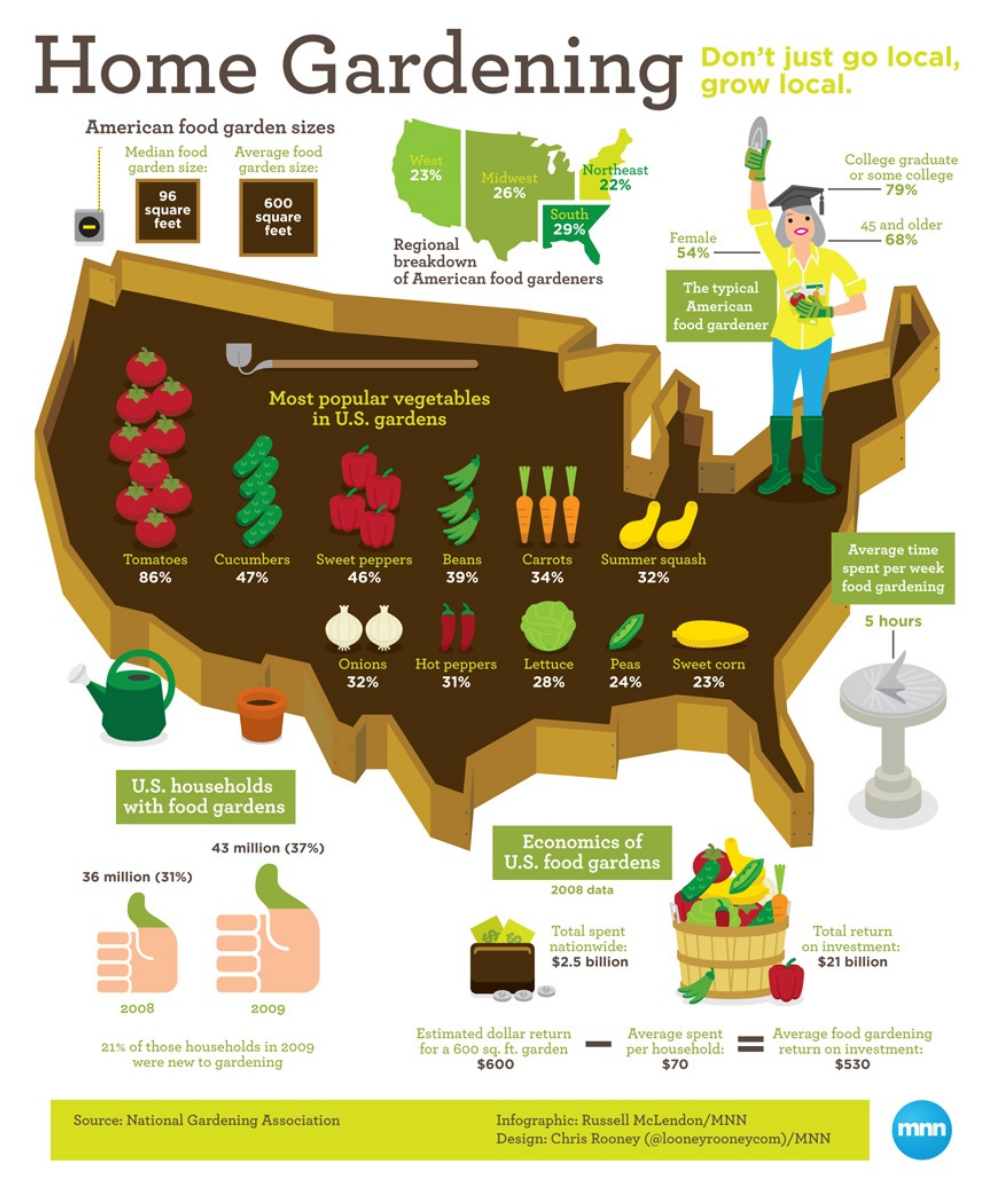


Figure 3.4: American home vegetable gardeners characteristics and activities. These customers are mainly college educated, 45 and older, and are located all over the United States.

Our second supplementary market is indoor growers of medicinal herbs (Fig.3.3 b). We are primarily

interested in this market segment because these customers have the ability to pay much higher prices for our product than other segments. This would help us accomplish our mission of reducing food insecurity by serving as a highly profitable market segment, perhaps allowing us to sustainably sell products to our target markets for very low prices, or even at a net loss. Indoor growers usually use a great deal of climate control equipment including fans, irrigation pumps, lights, air conditioners, and sometimes heaters. Often growers have 5 to 10 independent grow-rooms and need to control 40 or more devices independently. They would value our systems ability to expand and control as many devices as needed and the ability to separately control as many different climate zones as needed. Controlling many high-wattage devices is much easier with our system because we can control power coming from many different outlets. Real-time monitoring and control from a smartphone would help reduce anxiety about growing conditions and ensure that their plants are growing in the best possible conditions.

There are 300,000 to 600,000 indoor medicine growers in the US and that market is increasing rapidly, at 64% per year. They are especially located in California, Washington, and Colorado. They often are able to make large upfront investments to start growing. A basic growing operation of a typical size typically uses over $15,000 of equipment. These customers usually use 5 to 10 independent grow room controllers which each cost about $500. In this scenario, our system could offer similar value to that of $2500 to $5000 in existing climate control equipment, but with added benefits of wireless setup and control. If we pursue this market it would probably have its own product offering with targeted features and branding. Customers in this segment would probably want advanced features such as CO2 monitoring or security monitoring and alerts.

3.2.4 Assumptions and Testing

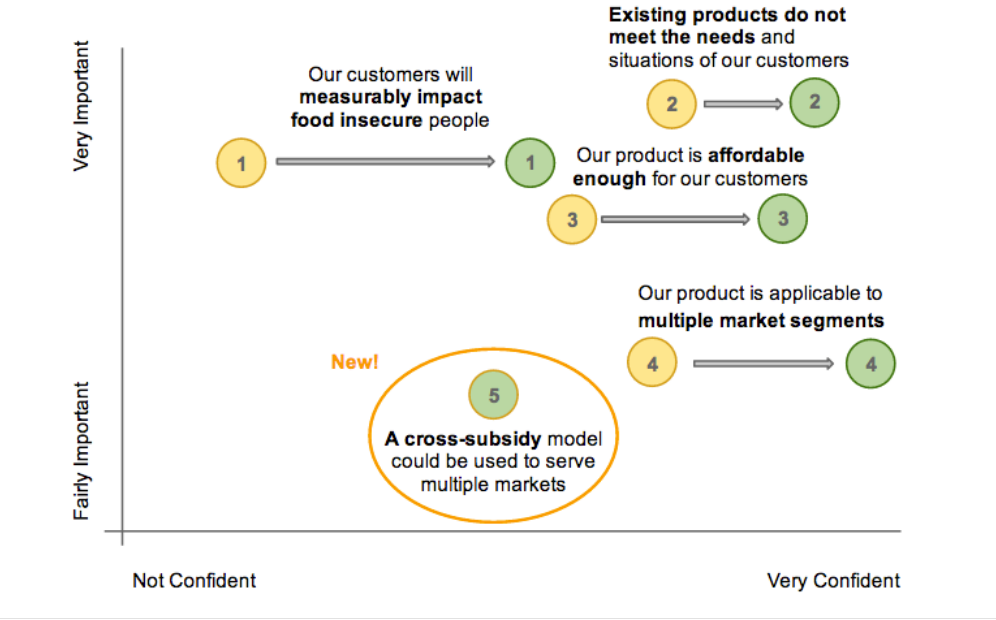


Figure 3.5: Key market-related assumptions and movements.

For this component we had five key assumptions (Fig.3.5). This section describes each assumption, illuminates its importance to our venture, specifies how we tested it, and explains our most-current understanding of it based on the results of our tests.

Assumption 1: Customers will measurably impact food insecure people of Massachusetts. We wanted to have more confidence in our claim that our product could indirectly provide more food for food insecure people. After talking to about 10 farmers and visiting farmers markets in low income neighborhoods like Dorchester, MA, we understood this connection more precisely.

We found that many farmers dont directly cater to food-insecure people, but 5% to 20% of their customers pay with food stamps. Some farms donate the extra food they produce to food banks. Some farms specifically choose to lower their prices and go to low-income farmers markets to have a positive impact on that community, and some farmers are food insecure people who grow for themselves. With this assumption were also implying the assumption that our product will improve the relationship between farmers and food insecure people. Hopefully our product will enable more farms to sell food to food-insecure people and boost the capabilities of farms that are already helping. We are fairly confident that this is also true. Most farmers we have talked to predict that with automated propagation, they could grow more food for less money.

We have concluded that our target-customers do impact the food-insecure, and that SNAP (food stamp) money is a useful metric for understanding that connection. We would like to have more direct evidence that our target-customers would be able to have more impact on the food insecure as a result of using our product.

Assumption 2: Existing products do not meet the needs and situations of our customers. We wanted to identify the automated climate-control systems currently used in agriculture and understand how our core market segments have responded to them, looking for frustrations and incompatibilities, as well as most-admired features. We want to understand the reasons why farmers do or do not use these automated climate-control systems. After to talking to 12 farmers, visiting 5 farms, visiting a climate-control-system vendor, and researching comparable systems, we had a much better understanding of the situation.

We found that our customers did not want to (or couldnt) pay the $700 to $2000 for a controller. Some farmers dont have the automatable equipment that a controller could control (they may water by hand, heat with a woodstove, or roll up and down the sides of a greenhouse manually). Farmers didnt want a permanently installed system requiring installation by an electrician. They use their greenhouses for many different tasks throughout the year (propagating on tables, growing in the ground, storing produce, storing equipment, housing chickens, etc.), and having a permanently wired system would be a nuisance when switching between greenhouse modes. Most farmers dont have a wired internet connection or a computer in the places where theyre growing, and some dont even have electricity.

We have concluded that our target farms do have climate control needs that are unmet by the products currently on the market or the farms are unable to afford a system that has the capabilities they require. We would like our product to fulfil this need for a highly functional, highly affordable climate controller and we suspect that it will also be well received by larger, more mainstream markets as well.

Assumption 3: Our product is affordable enough for our customers. Currently, many farms are not using climate-controllers simply because they can not afford the cost of buying them. We wanted to know how our price targets fit with the willingness to pay, as well as the ability to pay of our core market segments. We ran similar tests to those for the previous assumptions: talking to 12 farmers, visiting 5 farms, and visiting

a climate-control-system vendor.

We found that nearly all farmers invest in some climate-related equipment in the $100 to $200 range (industrial thermostats, timers, wireless temperature sensors). Every farmer we talked to said that they would buy our system if it cost $100. The hydroponics equipment retailer said she could easily sell our system for $700 (as long as it passes their tests, is all UL listed, and we can offer a 2-year warranty). She said that web-based data-logging would be worth $250 if we added it to one of her current control systems. One large-Established Farm owner, Scott Hurwitz of Silverbrook Farm, said a system like this would save him $100,000 a year. (But he currently has his farm-workers doing those tasks instead of an automated system. This apparent contradiction is a question we hope to investigate further next semester.)

We have concluded that a $150 sale price for a basic system is affordable to the vast majority of our target customers. We would like more confidence in this conclusion and hope to do more detailed willingness-to-pay studies next semester. This sale-price seems technically feasible given that our bill-of-materials for each product costs us about $65 (at quantity 500).

Assumption 4: Our product is applicable to multiple market segments. This assumption is not critical to our success, but it informs product feature decisions and targeting and marketing. We found that our system meets a set of needs that are very widespread. Like thermostats or timers, our system could control anything that can plug into it. We have features (like wireless setup, Internet-linked monitoring, and simple, near-infinite expandability) that appeal to many market segments: Large farms, small farms, outdoor growers, indoor growers, experts, and beginners.

We have concluded that we would like to design our product to work especially well for two core market segments, yet keep our product versatile enough that it would appeal to a wider variety of markets.

Assumption 5: A cross-subsidy model could be used to serve multiple markets at affordable prices. This was a new assumption that came out of our results from assumptions 3 and 4. We think we could sell systems to some segments at a large profit margin and use that money to subsidize the cost for customers segments that have very limited resources. We dont have enough information to come to a conclusion about this yet, but we will probably do an initial product launch with a single product at a single price, then consider the utility of a cross-subsidy model for reaching especially-financially-constrained segments.

3.3 Refinement

After identifying the three core market segments we are interested in, we have decided to concentrate on meeting the needs of two of our segments: Beginning Farms and Social Farms. Weve found that the Established Farms are more likely to be able to afford existing systems. Beginning Farms and Social Farms have the most unmet needs and also have the greatest impact on the food insecure.

Going forward, our plan is to do most of our co-design and productizing with Beginning Farms and Social farms. This will inform features we decide to emphasize or take out, our decisions for distribution and service, and our branding and marketing. We do, however still believe that our product, while optimized for these two market segments, would be well received and sought after by our other market segments. To keep our market options open and flexible we want to choose product features and branding that would be acceptable to a broader market. It will be a product that is optimized for Beginning Farms and Social Farms, but is versatile enough to be be enjoyed by many. Later in the future, it will probably make sense to

have a product line with several product levels and service levels, so that each market segment can have a product that has their own optimal user experience.

3.4 Conclusions

We have identified our three core market segments: Established Local For-Profit Farms, Nonprofit Farms with a Social Mission, and Beginning Farms. We estimate that there are about 1350 of these farms in Massachusetts. From those numbers we predicted that there about 390,000 of these farms in the United States, but new information is showing that our number is probably between 100,000 and 200,000 farms in the US. These farms currently impact food insecure people and we have confidence that our product will better enable them to put more food in the hands of food-insecure people. We have found that current climate-control products on the market do not meet the needs of these farms at a cost they can afford, especially for Social farms and Beginning Farms. This is where we fit in.

We are also interested in two other markets that could supplement our customer base, helping us reach the volumes of scale that could support full-time development engineers and support networks. These markets are Home Gardening Enthusiasts and Indoor Medicinal Herb Growers. Indoor Growers is a very high-margin market that is rapidly expanding. At about 300,000 customers, it would double or triple the potential customer base. Gardening Enthusiasts in a huge market at about 28 million customers. It would expand our customer base one-hundred-fold.

Our short-term direction for development is to focus on designing for the needs of Beginning Farms and Social Farms and initially come to market with a single (expandable) product offering, that is versatile enough to appeal to many market segments. Longer-term development would focus on creating a product line with several unique product-service offerings that all share our base platform technology, but would bring the optimal user-experience to each market segment.

4 Comparables and Prior Art

Massachusetts, USA, May 8th
Kelsey Schilperoort

4.1 Introduction

The overall goal of this venture is to assist the food insecure, those who do not have ready and consistent access to fresh produce throughout the year, living specifically within the state of Massachusetts. We focused our efforts on increasing food production yields for small farms through improving climate control for indoor seedling cultivation areas. With controlled environmental conditions such as temperature, light, and humidity, we hope to decrease seedling mortality, and thus increase farm yields for the subsequent growing season.

Previous research on existing climate control systems primarily centered on larger and more industrial farm control systems. However, as our team is targeting smaller farms as the primary users for our climate control system, these specialized and often customized industrial systems were not an adequate product comparison. Thus, for this semester, the Comparables and Prior Art component, which serves to analyze previous work and current products that may be of interest to our team, focused on researching current climate control systems which are marketed towards agricultural applications and which are designed for use on a smaller scale. Through researching current products on the market and in use in small farms around Massachusetts, desired characteristics of current systems were identified and compared to the product we are currently developing.

Research and comparison brought several product characteristics to light, some of which might be beneficial to introduce into our climate control system in the future. While our product improves upon many current climate control system problem areas, such as limited number of environmental actuators that the system can control, lack of data-logging, and high purchase cost, there are other characteristics that our product currently lacks. Characteristics that could be introduced are an alternative power source such as battery or solar powering, as well as waterproofing for our system such that if can be used in outdoor environments. Given additional user research in the future, desired characteristics of climate control systems could be ranked, and their relative desirability can be determined. From this, our product could be better shaped to what our users want, and unnecessary features can be avoided such that affordability can be optimized while still retaining functionality of the product.

4.2 Description

A variety of different existing climate control technologies were analyzed, and device characteristics were gleaned from this analysis. One example of existing products which are representative of the climate control market, and/or which contributed unique characteristics for future product modification of our system can be seen in Fig.4.1. All the compared systems are in Appendix D.

4.2.1 Intelligent Growing Systems CO2/RH/Temp Controller

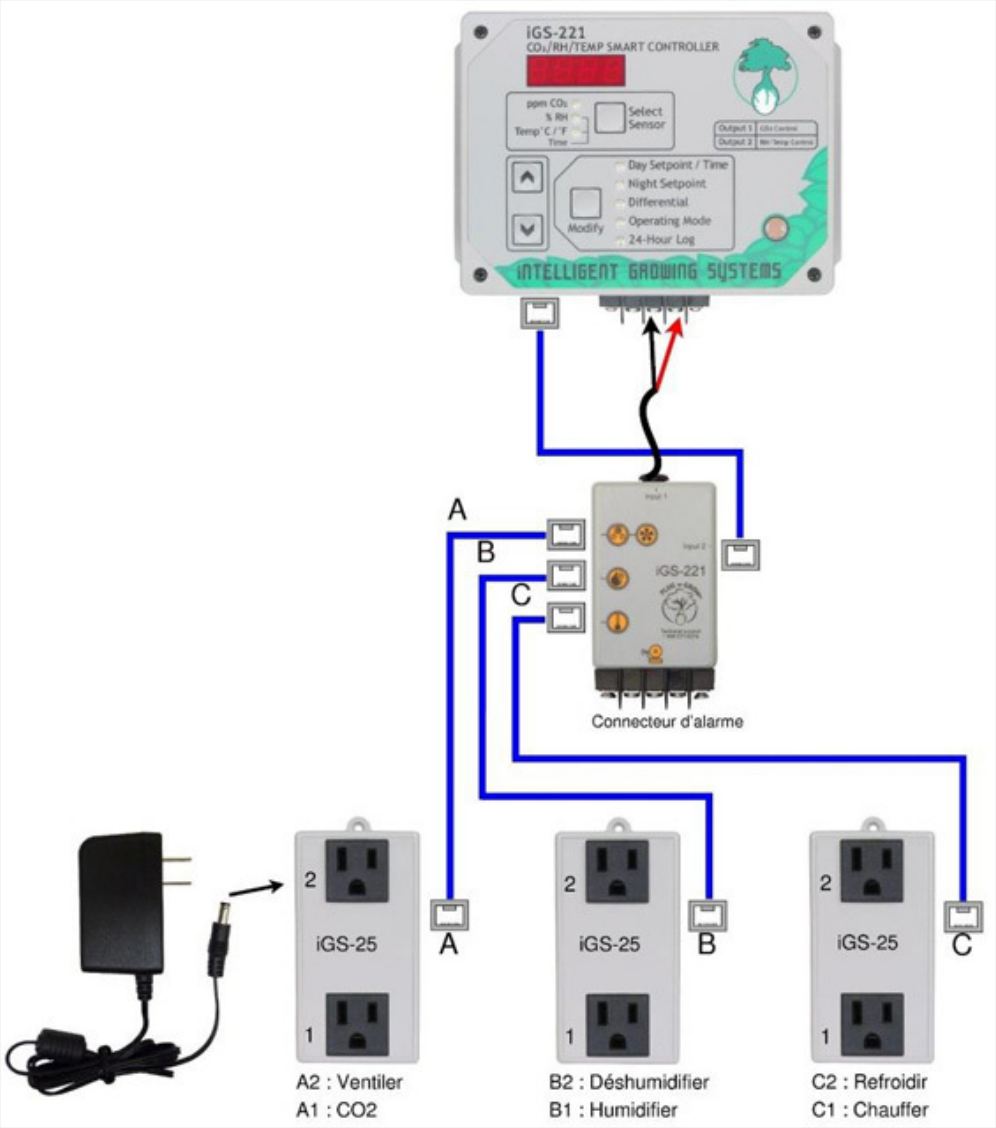


Figure 4.1: Intelligent Growing Systems: carbon dioxide (CO2), relative humidity(RH), and temperature controller.

Price: $785

Functions: Environmental control - 6 controllable outlets, control box, sensors - capability to control only parameters of CO2, relative humidity, and temperature

Downsides: Limited functionality, wired, price

Features Desirable to Our Product: Modular, automated, has limited data-logging capabilities, compatible with existing additional technologies(through use of standard electrical plug control)

4.3 Assumptions and Testing

Our team started by forming some basic assumptions concerning existing products that were similar to our own, as well as how our product would be adopted onto the market with similar competing products. These basic assumptions are as follows:

1. There are existing solutions that control the climates of greenhouses/propagation chambers We assumed that there were already marketed solutions for climate control in indoor areas, and that these solutions were accessible and currently being used by users such as small farms.

2. There is room for improvement in current technology In order for our product to be useful to users, as well as marketable, we assumed that there was room for improvement in current climate control systems.

3. The improvements we are making make the system better for our users We assumed that the improvements that we were making by introducing our product onto the market would influence our target group, the food insecure.

These assumptions were easily tested in simple $5 tests by internet searching and phone calling various farms in order to ascertain the state of climate control systems in use. Internet searches and research on hardware companies and companies geared towards farming applications also provided information as to the scope of products available on the market which are geared towards climate control. After gathering this initial information, $50 tests were performed in the form of onsite visits to local farms in Massachusetts. These visits provided additional information about current systems and what they are used to control on farms.

Based on these tests, we found that there is a wide spectrum of products both available and in use by individuals and farms for the purpose of controlling elements of the environment. Through our own assessment of current systems, as well as through interviews with climate control system users, we identified problem areas with current systems, confirming that current technology for climate control systems is not ideal. In order to prove or disprove the third assumption, more research should be conducted on how our product is perceived once it is deployed to a larger number of farms and individuals. Our team has been making efforts to increase the usability of our system, such that the interface is very easy to figure out, as well as use. Initial feedback on our product interface is promising, but further user feedback is warranted for future semesters.

Based on these tests and input from potential users of our system, we were able to compare current systems to our own and determine if our product system is bringing something new and useful to the market. Many of the existing systems have similar characteristics to our product. However, we found that none of the current systems on the market have the same range of capabilities as our system. Many of the existing systems have select characteristics that are very similar to our product, but none had all of the features that our product offers. We found that systems on the market included one or more of the following characteristics:

1. Durable
2. Modular
3. Compact
4. Automated
5. Simple user interface
6. Wireless control
7. Wireless sensing

However, we also found that current technology on the market had problems such as:

1. Limited indoor space that the system can control
2. Limited number of environmental features that the system can control
3. High purchase cost
4. Difficult to install
5. Non-intuitive control interface
6. Antiquated design and blocky appearance of interfaces
7. Not easily modifiable
8. Require a source of electricity
9. Limited or no data-logging capability

By comparing the many systems on the market, we found that they individually do not cover all the features that are needed in a climate control system, as well as collectively have very limited functionality. Systems are limited in how many actuators they can control, and thus systems have either a size limitation on the indoor space they can control, or a limited number of environmental elements that the system can control. Based on the numerous problem areas identified, there are clear improvements that can be made on currently-existing climate control systems. Our product brings something new and useful to the market in that our system combines several main attributes that have thus far been collectively absent in climate control systems.

Our climate control system is:

Affordable, Wirelessly Controlled, Automated, Expandable, Versatile, & Modular

Our system can be easily installed as it is wirelessly controlled and thus needs no additional internal electrical wiring (although it does require power from standard 110 volt outlets). Our system can also be easily controlled from off-site locations, due to its wireless capability. Most of the systems on the market were not affordable for our target users, or if affordable, had very limited functionality. While most of the current climate control systems are in the range of hundreds to thousands of dollars, the price point that would make our product affordable for our users is on the order of $100. Our system is therefore geared towards being the most cost efficient while still retaining the desired characteristics of current systems. An additional attribute of our system is that it is able to integrate some of the technology that is currently being used on farms, such as heating mats, humidifiers and fans. Our system can also be expanded for use in enclosed spaces of any size, due to its modularity, which consequently also allows for easy replacement of product components in the event of damage. There is effectively no limit to how many outlets our system can individually control, and even after product installation, the system can be expanded later on.

4.4 Refinement

Based on current product characteristics and possible improvements in current climate control systems, key characteristics that our product needs to include in order to compete on the current market were identified. Additionally, through interviews with potential users, the importance of low power consumption, or alternative power sourcing was identified. Several of the contacted farms did not have power outlets in regions where they would be growing seedlings. In order to improve our product system and increase the number of potential users that are able to use our system, alternative power such as solar could be explored. Moving forward in future semesters, research into user ranking of desirable product characteristics as well as ranking of problem areas in products should be undertaken. Through this additional research, trade-offs between different characteristics can be determined, and a more refined point as to which features our users are willing to forgo in the interests of economy could be determined. Additional features could also be integrated into our system, such as waterproofing, which is fairly standard for climate control systems geared towards plant-growers.

4.5 Conclusions

1. Based on the insights from current products, as well as input from farm owners and other potential users, the following could be done in the future in order to create a more useable and highly desired climate control system.
2. Perform user research and obtain a ranked list of desirable and undesirable climate control system characteristics
3. Research alternative methods of system powering, such as battery power or solar power
4. Research waterproofing for our product system, such that it can be expanded for use in outdoor environments

5. Perform tests on the durability of our system, obtaining information on how long the product would last before it breaks down, and what conditions would cause our system to break down

5 Product-Service System

Massachusetts, USA, May 8th, 2014
Ian Daniher

5.1 Introduction

The product system for ADE-MA consists of a variety of commercially available, off-the-shelf components communicating wirelessly to monitor and control parameters of interest in small-scale agriculture. Environmental parameters such as temperature and humidity can have a profound impact upon the health of vegetables, especially in the early stages of growth. We found affordable solutions for monitoring these parameters with exceptional battery life, reasonable accuracy, and long range. Currently owned or easily available off the shelf 120V equipment such as pumps, space heaters, and blower fans can be switched on and off by remote control outlets using wireless technology similar to the previously described sensors. An affordable dual-core computer, designed for smart TV use, runs a system of software components built using Rust and Node to provide monitoring, regulation, and user interface.

5.2 Description

5.2.1 Hardware

Our system currently offers control of devices using up to 10 Amps of current at mains voltage, measurement of air and soil temperature and moisture, and a flexible way of building control algorithms.

Figure 5.1: Hardware components of a SproutWave system.

A small computer with a broadband radio receiver runs software to extract information from wirelessly transmitted digital packets, automatically sent at regular intervals from off-the-shelf sensor hardware. A USB transmitter allows software to make environmental changes by switching on and off heaters, fans, pumps, and lights attached to RC outlets to keep plants in their best environment. The computer provides a remote user with the ability to view logs and remotely control the growing environment.

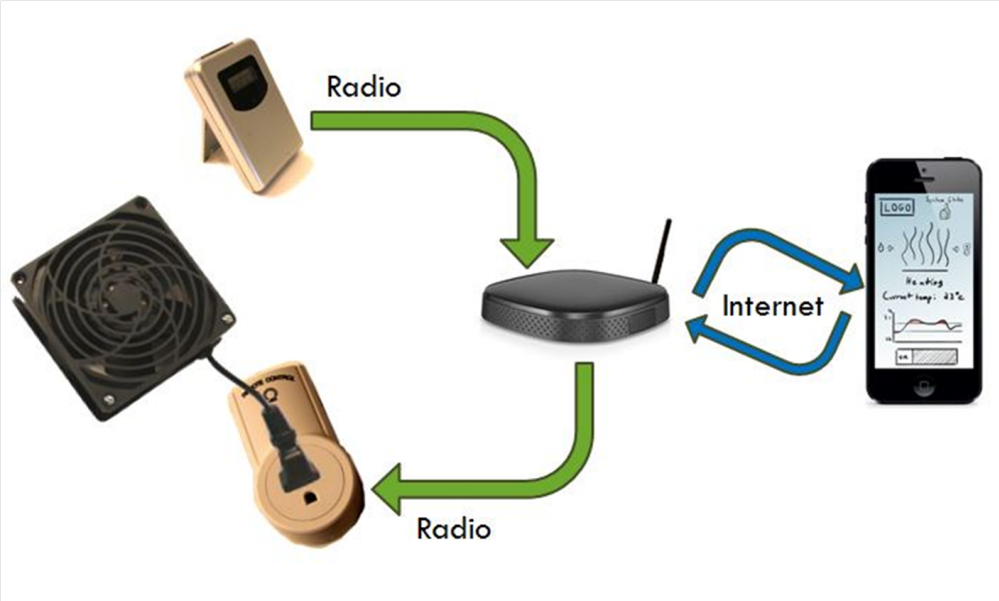


Figure 5.2: System parts and communications.

Our system uses USB radio receivers supporting the rtl-sdr software library to listen to radiowaves around 434MHz, a globally used industrial, scientific, and medical (ISM) band. All of our remote sensors and switches work in this frequency range, which also overlaps with the amateur radio 70cm band. To further enhance the functionality of the system, the hardware receiver can also be tuned from 24MHz to 2GHz, giving direct access to meteorological transmissions and public utility equipment.

The transmitter solutions for our system serve to transmit a 6000 pulse per second packet at 433.92MHz to communicate with ETekCity brand outlets. A USB microcontroller (the atxmega32a4u) sits between a computer running our control code and a radio frequency oscillator / transmitter. With an off-the-shelf-parts-cost of approximately $10 at quantity one, its not expensive, but requires custom wiring and offers extremely limited functionality.

Due to the efficiency of the decoding software, the computational requirements for this project are relatively minor. Many different types of small embedded linux computers have been successfully tested, including variants of MK908-style thumbdrive computers using the quad-core 1.6GHz RK3188 CPU and featuring 2gb of RAM, the Freescale WandBoard Quad, and several platforms using the A20 family system-on-a-chip made by Allwinner. The A20 is a high-feature-density all-in-one solution for a variety of mobile devices. It offers the PC-standard interfaces of HDMI, component video, audio IO, high speed USB, with embedded processors including microSD, SPI, I2C, additional serial ports, and a friendly hardware configuration method schema with thorough Linux kernel support. At the end of Spring 2014, the SDK-758 made by Sunchip tech out of Shenzhen seems to be a high quality product working well for our needs. Links to complete documentation for the A20 and SDK758 are included in the Technical Appendices.

One of the benefits of a software-driven communication scheme is the ability to support multiple different families of remote sensors. Any wireless sensor that uses the standard 434MHz band, and many that dont, can be easily integrated into our system. A complete list of protocols and hardware with support of varying levels of sophistication is included in the Technical Appendices, but the system currently supports two ideal sensors for air and soil temperature and moisture. Both are readily available online and communicate using similar pulse-width modulation protocols, where a long pulse represents a binary 1 and a short pulse represents a binary 0. The sensors represent temperature as a signed 12 bit number containing a measurement in units of 0.1 degrees celsius. Humidity is represented an 8 bit number with units of 1% RH. Accuracy is typically better than 2 degrees celsius and 3% relative humidity.

The outlets (Fig.**??**) used in our system work on both 120 and 240VAC mains voltage and . At a maximum power of roughly one kilowatt per $7 outlet, they provide excellent power handling density in an Underwriters Laboratory certified product capable of handling upwards of one million actuation cycles for a twenty year continuous use lifespan at a ten minute click rate. These outlets are available in preprogrammed sets of one, three, or five, typically with one or two batteries-included remote controls. The outlets are sequentially coded per lot, with a four digit unique lot code. Each outlet can be individually turned either off or on by default, with multiple outlets able to be easily keyed for the same codes. A 25 bit digital signal is transmitted five successive times to switch an electromechanical relay. A minimum of digital glue circuitry sits between a radio receiver and power handling circuitry.

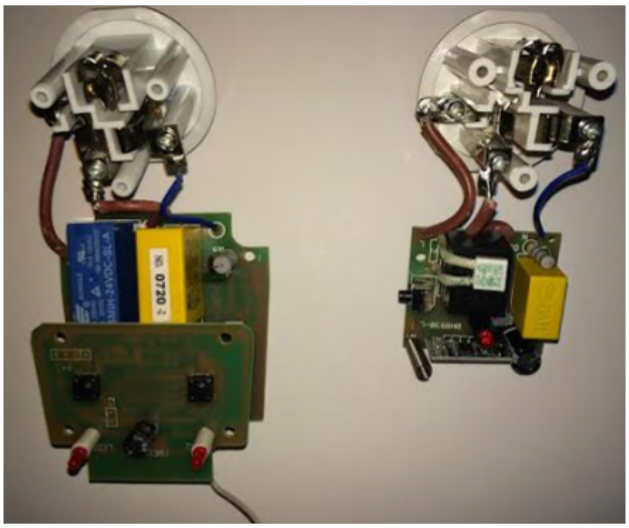


Figure 5.3: Two models of ETekCity outlets. The larger offers buttons for on and off, but does not support absolute commands. Note the yellow supply capacitors and the black/blue relays.

5.2.2 Software

The process of translating from analog measurements of radio frequency energy to independently useful measurements in human-readable units is a nontrivial technical problem. Both significant amounts of digital signal processing and parsing logic are required to extract a sensor transmission packet from a high speed analog signal. This signal processing, demodulation, and decoding logic need to not only operate continuously, as there is no synchronization between remote sensor nodes, but must also be efficient enough to run at high sample rates to cover multiple varieties of sensors, each at their own frequency, on affordable commodity hardware. A Kahn Process Network (KPN) paradigm (see Technical Appendices) offers guarantees about system stability and a convenient framework for building parallel applications, but no suitable implementations existed. Early prototypes of our control code, constructed and refined in the first part of the semester, used the LibRedio and LibOut Rust libraries to carry out this logic, but were of limited efficiency due to dynamic typing, and offered a terrible software interface for constructing even relatively simple data processing flowgraphs. To facilitate the efficient description and construction of graphs of the many interconnected processes required to carry out the necessary information manipulations, a small programming language and compiler was built. This language, RatPak, and the control software built in it, are both documented in the Technical Appendices. RatPak builds upon LibRedio and provides a high level system interface to both input and output abstractions. The end product is compilable Rust-language source code, which produces a small executable binary that when run, spawns a system process for each software module, reads information streams from the receiver, produces a stream of sensor readings and outlet commands, and can send commands to remote outlets. This binary can be compiled for any variety of computer system - Windows, OSX, or the Debian embedded Linux running on our current A20 hardware.

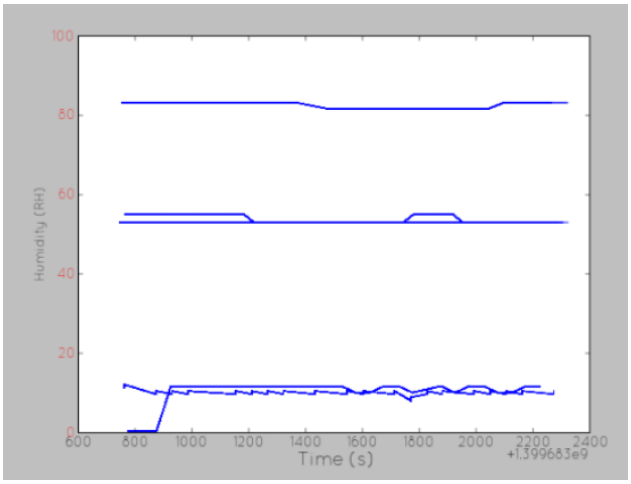


Figure 5.4: Graphing prototype, code in Technical Appendices, showing multiple sensors over a 10m period.

The user interface has gone through several complete revisions, but in all cases consists of a set of scripts capable of accepting data and user input, logging data to a database, drawing it as a plot, and providing this information to a user. Earlier this semester, a Python prototype was designed and partially implemented using the Tornado web framework, but was largely abandoned for want of sustained attention. A second, more minimalist prototype was assembled for quick demonstrations, but Pythons synchronous nature makes the dual responsibilities of reading data and providing plots to a user a challenge. User interface design and development remains a wide open area, with substantial amounts of exciting work yet to be done.

One promising opportunity for our user interface involves a collaboration with Apitronics, an open source agricultural monitoring device design company. A $20k kickstarter campaign funded their development of Hive, a Node application providing plotting, exploration, and status alerts for remote users. Apitronics currently supports this software system for use with a full feature weather station and a feature-rich remote monitoring platform based around custom boards using XBee wireless modules and an ATXMEGA microcontroller. In their system, a small linux computer, the Hive (BeagleBone Black) connects remote nodes called Bees to a web user interface. The parallel system architectures, coupled with their platforms relative maturity, makes it an attractive system from which we could develop our frontend, or, perhaps even codesign into a universal software platform for agricultural monitoring and control applications.

5.2.3 Wireless Backbone

One of the weaknesses of our preliminary system architecture is the dependency on a backbone (internet) connection for providing remote access to environmental data and remote control of connected hardware. While 70% of farms in the United States have internet connectivity, it is not uncommon for farms to have no reliable connectivity in propagation spaces, let alone the field. At SOS, internet is available, but distance and copious amounts of cement make standard 802.11 WiFi at 2.4GHz/5GHz unusable. To this end, we initially procured a CradlePoint LTE cellular modem, loaned complete with service plan by Verizon.

Figure 5.5: CradlePoint LTE cellular modem, loaned complete with service plan by Verizon.

To our surprise, the reliability of this system, as well as the complexity associated with exposing an information stream from a device behind a network-address-translation (NAT) interface rendered this approach challenging. From a long-term view, cellular technologys substantial recurring costs further make it an unappealing connectivity solution for our system. Several possibilities exist addressing this need for robust, affordable, low-speed, longer-range communication. Wireless transceivers with 100-1000x the power output of the transmitter used for outlet control are available for $20-$40 at quantity one in the form of handheld walky-talky style radios. With proper software for modulation and demodulation, ranges up to hundreds of miles can be achieved - theoretical limits allow routine operation at 30dB below the noise floor (-30dB SNR). Using less extravagant protocols, messages can be piggybacked on existing wireless communication infrastructure, such as that routinely used for the tens of thousands of small weather reporting stations scattered across the US. In either case, the venture could establish a limited amount of specialized hardware in an arbitrary location and provide support for hundreds of remote sites hundreds of miles in any direction, providing an internet-accessible user interface without requiring an internet connection proximate to the growing site. Several possible protocols, their relative advantages and disadvantages, and additional resources are documented in the Technical Appendices.

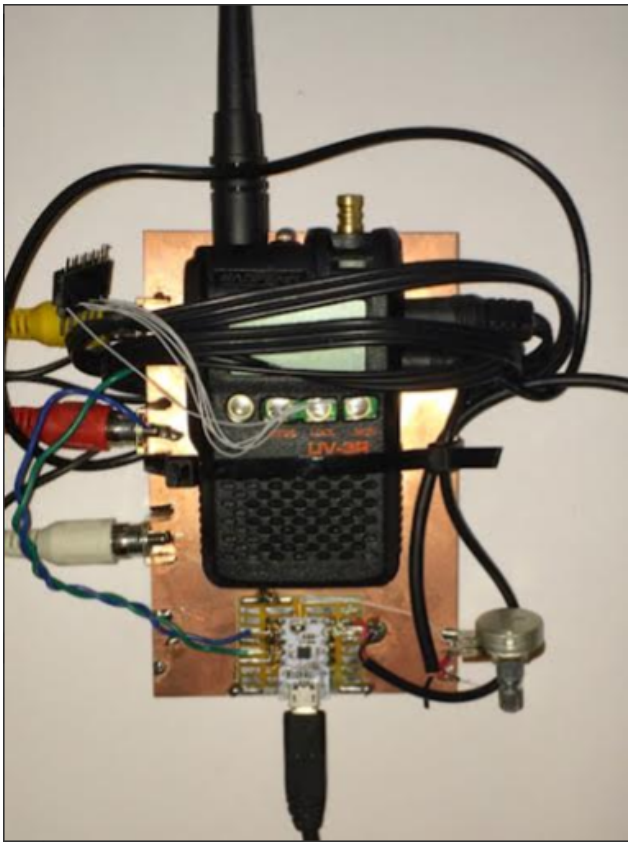


Figure 5.6: Baofeng UV3R Handheld - Prototyping System.

5.3 Assumptions and Testing

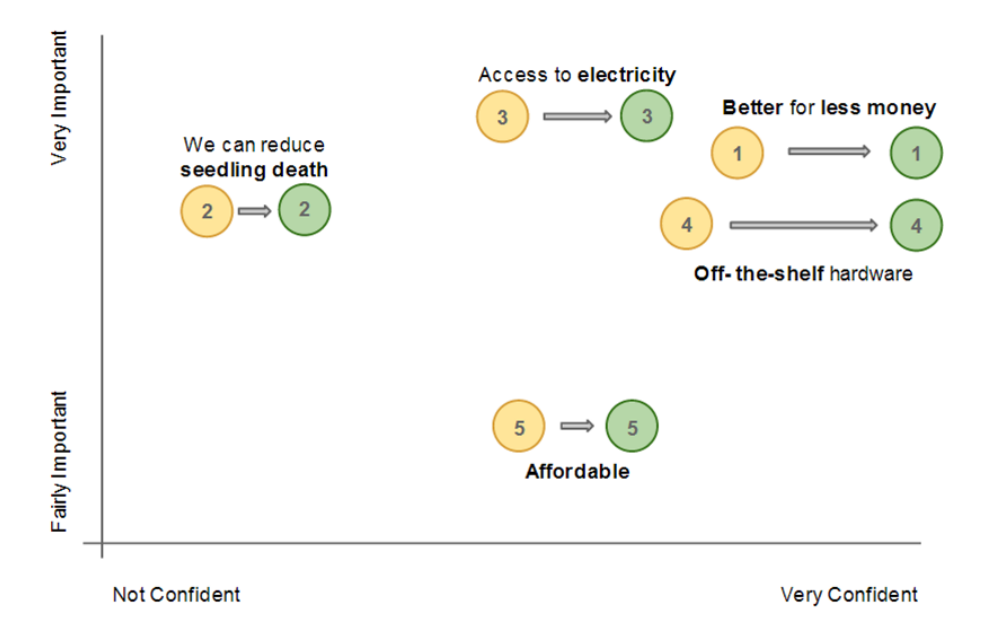


Figure 5.7: All assumptions related to the product/service system, graphed according to their importance and our confidence in them, and their change from the beginning of the semester to the end.

The technical team concentrated on testing assumptions 4, and 5, Our system can be constructed exclusively from off-the-shelf commercially available hardware. and Our product will cost a small enough amount of money that our intended user can purchase it. To do so, we developed 4 more specific assumptions (Fig.5.8

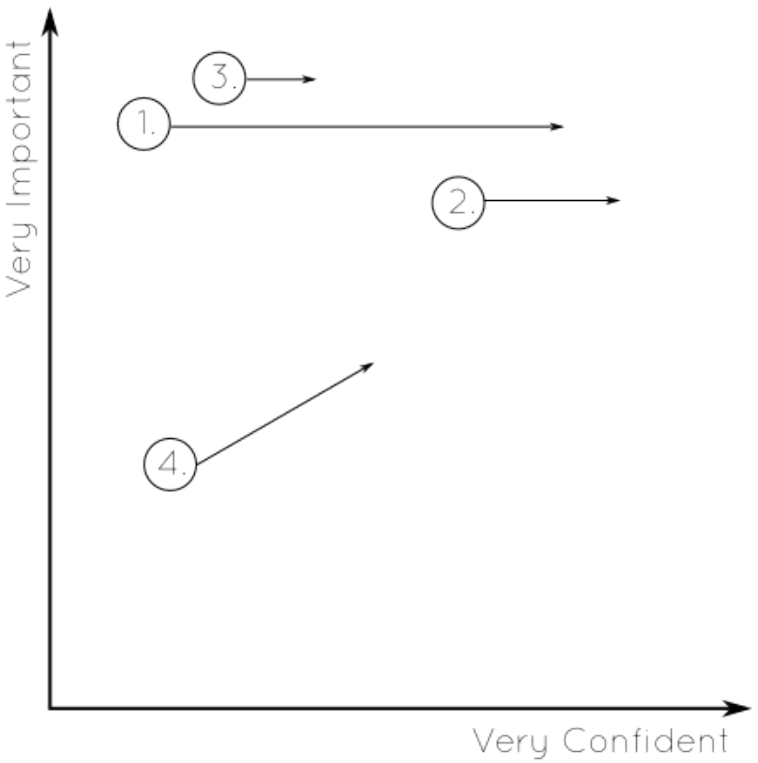


Figure 5.8: All assumptions related to the product/service system, graphed according to their importance and our confidence in them, and their change from the beginning of the semester to the end.

1. Our system can reliably decode and record information sent from a variety of environmental sensors. This semester a fairly sophisticated backend was developed capable of robustly logging information from both soil and air sensors.

2. Our system can reliably control remote outlets. We started the semester with prototypes suggesting this, and these prototypes were refined into a comprehensive backend with a separate database mapping outlet names to codes.

3. Our system has is useful to and usable by farmers. This is the most important assumption made, but the majority of work this semester went into establishing a backend robust enough for future development efforts.

4. Our system has a low operational cost. Before this semester, internet access was taken for granted, and cellular tech was the assumed fallback. After field trials, internet access was determined not to be universal, and cellular technology proved to be less ideal than expected. Research and early experiments have suggested alternative solutions, but more work is required.

5.4 Conclusions

With a solid backend, exciting directions for future refinement, promising prospects for user interface, and a proven hardware platform, the product-service system is evolving quickly. The work this semester served predominantly to establish a technological base for developing the full SproutWave product system and validate critical assumptions with regards to feasibility. With an additional semester of software development, I expect that we will be able to provide a capable, affordable, and usable agricultural automation system to improve farmer quality of life and labor efficiency.

6 Value Proposition

Massachusetts, USA, May 8th, 2014
Colby Sato

6.1 Introduction

A person is food insecure if they are at risk for malnutrition or starvation. 11% of MA is at risk, and one in five Boston residents is food insecure. Our theory of change, which has been proposed by food policy experts, is that by increasing the production of food at local farms, more fresh, healthy food will reach the mouths of the food insecure. From conversations with farmers, we have learned that a limiting factor of food production, and a source of financial loss and personal anxiety is seedling propagation. Despite investing hours each day maintaining growing conditions inside greenhouses and other growing spaces, farmers regularly lose between 20% and 50% of their seedlings. We aim to change this experience with SproutWave, an autonomous climate control system that enables farmers to remotely monitor and control growing conditions. In this section, the various ways in which SproutWave offers value will be discussed and future areas of work will conclude.

6.2 Description

Propagation is the process of growing plants from seed to a plant large and strong enough to be planted in the earth. Seedlings easily perish from mold, exposure to air or soil that is too cold or warm, insects, or from insufficient watering. An example of temperature fluctuations can be seen in Fig.6.1. With all these factors at hand, farmers regularly lose between 20% and 50% of their seedlings[1]one year, a farmer lost 90% of her seedlings to mold. A farmer we spoke to, Kim Almeida of Blue Vervain Farm, described seedlings as infants on their deathbed.

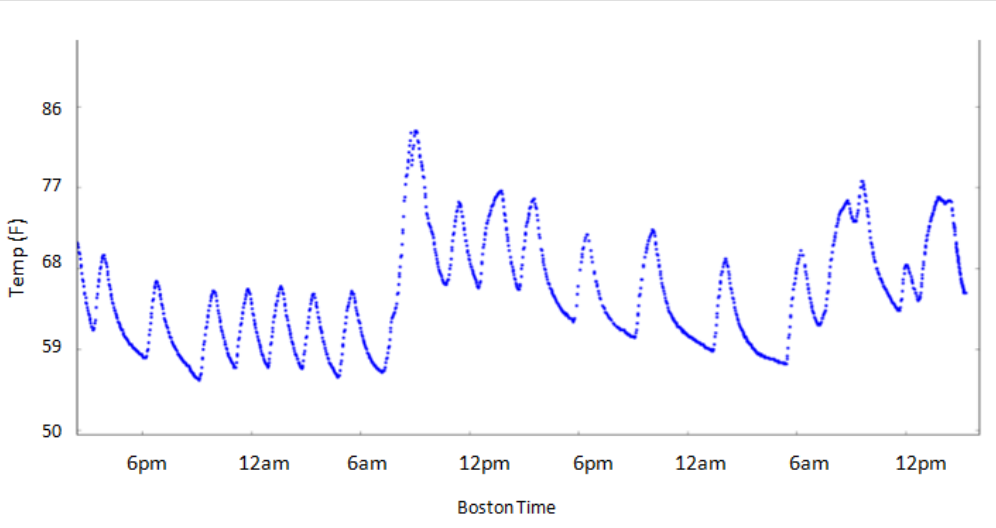


Figure 6.1: Temperature fluctuations such as these are regularly experienced by seedlings at SOS and contribute to high seedling loss.

Any Value Proposition is a subject to change as the product is adjusted and more information is acquired and processed. We recognize that the precise value of SproutWave in dollars and hours for different farmers is not yet fully understood, but we are nevertheless certain that this system provides significant value for farmers that propagate seedlings in greenhouses or other indoor areas.

6.3 Value Proposition Statement

The seedling propagation greenhouse is one of the highest concentrations of value on a farm. As such, farmers invest significant labor, resources, and mental energy managing the climate inside their propagation space. Along with this, farmers carry anxiety surrounding the risk of propagation. SproutWave aims to change this experience, reducing the labor required to control climates and reducing the anxiety and uncertainty of propagation through remote monitoring and automatic control. Not only does SproutWave offer great value, but it is also affordable at $150 for a starting unit, which is the price of other competitors. Affordable and powerful, SproutWave carries sprouts from seed to the field.

6.4 Components of the Value Proposition

6.4.1 Simplification of Farmers Lives

As stated before, farms regularly lose between 20% and 50% of seedlings. According to one farmer we spoke to, Scott Hurwitz, most of this loss happens because farmers are too busy to monitor and control every aspect of the plants environment, and baby plants are naturally incredibly vulnerable. To maintain the environment in greenhouses, farmers need to walk to their greenhouse, feel the temperature and moisture, then adjust fans, heating pads, watering systems, or sometimes even crank open vents in the top of the greenhouse. This process breaks the flow of work a farmer might be doing out in the field.

6.4.2 Comparison With Other Systems

Even on farms with climate control systems, farmers can lose seedlings due to user error.[2]. Other comparable systems cost in the range of $600 to $785 and have much less functionality than SproutWave. For exact prices, see the Comparables and Prior Art section. We are currently developing the user interface, and once we have passed that through several iterations in codesign, we are confident that we will have a system that is not only increase yield, but also be designed such that it will remove complexity from the lives of farmers, thus reducing the anxiety of growing seedlings.

Current systems offering control of 120vac wall outlets typically have a fixed number of switches, reducing scalability. As our system is wireless, SproutWave can control an essentially unlimited number of electrical devices - adding new inputs and outputs is just a software download away.For this reason, SproutWave also offers incredible scalability at an affordable price. To add three more outlets and two more environmental sensors, one simply needs to invest 30 more dollars in the system, whereas other systems such as the Koubachi cost $99/set of sensors[3]. Comparisons on price and functionality can be found in the Comparables and Prior Art section.

6.4.3 Reduction of Labor

Some farms have people employed all day, every day of the week, monitoring the greenhouse. Even with people paid to watch the plants all day, when no one is at the farm, or farmers are in the field, there is concern about the state of the greenhouse, and in this way, seedlings are a major source of stress for farmers. Current systems that aim to solve this problem are expensive and therefore inaccessible to farmers with limited cash. In fact, some farmers struggle so much that they themselves are part of the food-insecure population.

6.4.4 Reduction of Anxiety

Conditions in the greenhouse are such a large concern for farmers that even while away from the greenhouse such as in the field or at home, they worry about the health of plants. SproutWave will allow farmers to check in on the growing conditions in their greenhouses via a mobile application. This increases feelings of control and connection to plants. Being able to control the climate remotely also means one does not need to walk or drive to the greenhouse every time the weather changes in a way that could kill seedlings.

6.5 Assumptions and Testing

Assumption 1: Increasing yield during propagation would enable farmers to extend their season. This assumption is not critical to our success but if proven true, it would mean our system could lead to more food production earlier in the year, when not much local food is for sale.

This assumption had already been tested in past semesters, but we learned more about it this semester. We tested this assumption by visiting farmers at farmers markets and talking with farmers over the phone when they were less busy with customers. We learned from Catalina, the farm manager at Serving Ourselves (SOS), that farmers who plant crops outside of the greenhouse are limited by the weather outside. The climate control system cannot change when farmers can plant outside, but SproutWave can significantly improve the odds of having seedlings ready when outdoor conditions are good for planting.

Propagation loss is a significant challenge farmers face every season, and slows the planting of crops. It is not uncommon for the first few waves of seedlings to be not entirely full. The fields eventually fill, but it can take a few weeks to fill the fields, and this lost time is lost money. Future work on this assumption would look into calculating how much more money farmers could earn if their first few waves of seedlings had losses reduced to the average rate of seedling loss with SproutWave. This average rate will be measured in pilot studies in future semesters, and will require a coordinated effort between the technical team and the point person on Social Benefit Analysis.

Assumption 2: The money farmers save from use of SproutWave would pay for the initial cost of the system.

We have asked farmers how much money would be lost if an entire greenhouse of plants were lost and what is the value of a climate control system. Initial findings are that in a single greenhouse, owned by Scott Hurwitz, an established farmer, several hundred dollars of seedlings may be propagating at a single time, so the loss of that many seedlings easily pays for the system. However, Mr. Hurwitzs greenhouse was one case, and it will need to be investigated what is the average loss in seedlings for greenhouses of various sizes. This would provide a quick way to calculate the value SproutWave could offer farmers. This calculation could be factored into the Social Benefit Analysis, but it also has implications for the business model and value proposition. With this information, not only could we communicate the value of SproutWave better, but farmers could make more informed decisions, and we can gauge a fair price.

Also worth calculating is the cost of labor invested in the seedlings lost. When seedlings die, more seedlings need to be maintained to fill the holes the seedlings left, or more seedlings are grown than there are holes in the ground with the expectation of seedling loss. If fewer seedlings were lost, farmers would not need to grow as many seedlings. After pilot studies validate an average rate of survival or a range of survival rates, we can calculate the labor hours saved by not needing to maintain additional seedlings. Being able to calculate the cost of labor saved by reducing seedling loss, farmers could see more precisely the value SproutWave would offer them, and if we apply for grant money, we can include these calculations is Social Benefit Analysis.

In user interviews, we found that seedling loss also has a personal cost on the part of the farmers. Farmers have expressed how much they worry about the health of their seedlings. Having a remote monitoring system would allow farmers to worry less when away from the greenhouse. They could check on the health of their plants when they are out in the fields, working, or while at home with their families. Farmers that are new to farming also often live more than twenty minutes away from their farm, as many of them lease land, so it is inconvenient to drive to the farm to check on the health of plants.

Not only does SproutWave save farmers money through reduction of risk and seedling loss, it also removes anxiety from the lives of farmers and keeps farming from interfering with their personal lives.

Assumption 3: Our last area of inquiry was better framed as a question than an assumption. We wanted to figure out: what are farmers willing to pay for the system based on the value proposition we communicate? The distinction of the value proposition we communicate is included, because this assumption is also looking at how effective we are at communicating the value of the system.

To answer this question, we will share the climate control system with farmers and ask them how much they would pay for the system. The drawback to this approach is that they may state that they will pay more than they actually would. People generally avoid making other people feel bad, and they may see us

as college students and lower their stated price to make us feel good. We can keep this in mind and make it clear to them that being honest with us is good for us, and we will not be upset if they give us a low willingness to pay.

In an interview with Bianca, a local farmer, we saw the relationship between the value offered a farmer and the size of the farmers farm. Currently the farmer said she would appreciate it, because it would allow her to worry less. However, her farm is small enough that she can control the watering system, vents and heating manually, so she sees SproutWave as a luxury. Given this, when asked how much she would pay, she said she would pay $75. But if she tripled the size of her farm to have three greenhouses for propagation (total area 60x25 yards), she manually controlling the environment would be infeasible, so she would easily pay $200. Further research in this area will not only allow us to gauge the value of the benefits of this system, but also to clarify our target market.

The climate control system would be of significant value to farmers that have enough seeds propagating at any one time that manually controlling environmental actuators in infeasible. Based on the interview with Bianca, this size seems to be around 1,500 square yards. The size of propagation space at which manual control becomes infeasible will need to be tested further. Additionally, future tests need to look into how large this market segment is. Based on how large this market segment is, we will be able to make approximations for what price we should sell SproutWave at to be financially sustainable.

6.6 Refinement

When developing a value proposition, one must first determine what market segment one is focusing on. Then, based on the needs of that market segment and the benefits a product-service system offers one can develop a value proposition. As detailed in the Market Segmentation section, we initially had three market segments, but later we narrowed our focus to two market segments. We learned that beginning farms and farms with a social mission value affordability more, and farms in these market segments sell a higher percentage of their food to food-insecure people. We validated this decision by gauging the size of the of the markets. According to the EPA, there are 308,000 limited resource farms in the US. In an estimation, we arrived at there being 390,000 farms in our market segment in the US, and according to the USDA, there are 145,000 farms in the US that sell food directly to consumers. In Massachusetts, 20% of our farms grow produce, which gives the number of farms in our market segment at approximately 1350 in Massachusetts. These numbers should be further refined, but what they demonstrate is that this market is sizeable enough to focus on the value offered to them.

We had an idea of what would offer the most value, but it was not until we interviewed farmers in markets and over the phone that we learned about what our system offers. We initially did not know whether farmers wanted to control their greenhouse from a base station in their greenhouse, or whether they would prefer to use their phone. In interviews, farmers enthusiastically responded to the ability to monitor remotely, so we put development of mobile capabilities on the list of features to develop. We also reaffirmed the value of autonomous control of the climate, as this saves farmers significant labor hours and makes their workflow smoother. Next steps involve assessing what market segments benefit most from our system and are willing to pay enough to support our business. This will affect the business model and the product, service system.

6.7 Conclusions and Future Work

The more we talk to farmers, the more confident we are that seedling propagation is a significant pain point for farmers. Given this great need, we anticipated that farmers would be willing to pay $150 for SproutWave. This was found to be true to some extent. From initial testing, it seems that the financial and labor savings a farmer gains from SproutWave is tied to the size of their propagation space. Depending on the size of their propagation space, some farmers are willing to pay more than others. Another benefit previously considered to be minor is the ability to remotely monitor and control the climate. With the assurance that the system is autonomously controlling the climate and will notify them if conditions reach a danger point, farmers do not need to worry about their seedling when they are out in the fields or at home. This will have benefits the flow of their work and their personal life. Future Work

More research on this relationship would allow us to plot the relationship between these two factors and see more clearly the market segment that would benefit most from the system. Calculations based on information about farm size and prices gathered next semester should also help the team clarify this segment.

Comparisons of willingness to pay by farmers with calculated financial value (labor hours converted to dollars) would be interesting to see and might help us to identify whether we are selling the product short, overestimating the value of the system, or perhaps even failing at communication. Our ability to communicate value has a significant impact on business model, as different price points shape our profitability and sustainability as an independent company, so it would probably be good to have someone on the team working on advertising at the bridge between the Value Proposition and Business Model. Future work should look at the effectiveness of different ways of communicating the value of SproutWave

The value of products is often best expressed through stories of user experiences. With limited testing of SproutWave, we currently lack stories detailing the value of SproutWave. Once the system is deployed in pilot tests, it the business team should work closely with the user team to collect testimonials and document the financial and personal benefits of SproutWave.

Another area of high concentration of value on a farm is in food storage, and farmers lose produce from fluctuations in storage conditions, so the values offered to seedling propagation may apply to food storage. Egg incubation also requires exact growing conditions, so this is another area in which SproutWave could offer value. We are only beginning to understand the semesters, the value of SproutWave will become quantifiable and will allow more specific approximations for individual clients.

7 Business Model

Massachusetts, USA, May 8th, 2014
Rolando Chinchilla

7.1 Introduction

For the first time, ADE-Massachusetts started looking into potential business models to pursue in order to bring our climate control system (CCS) to the market and have a positive impact on the food insecurity issue in Massachusetts. Our main goal was to find a business model that would foster a profitable and sustainable business while increasing food security.

This component report describes the progress that was made on the business model throughout the semester. It starts with the decision-making process of potential business models followed by the chosen business model. Once a business model was chosen, several assumptions and tests were developed around it in order to assess its feasibility. Furthermore, it shows a business model canvas, financial projections, and a break-even analysis to test the feasibility of the project. Finally, it describes the next steps to take to launch the project out of the ADE pipeline and convert into a real business.

7.2 Description

Through a group ideation meeting, our team assessed over ten potential business models, from which four were selected to pursue based on feasibility.

The four options we were looking to pursue were:

1. Become part of the product line of an already established business (Grove or Apitronics)
2. Develop our own independent startup
3. Be part of a government program to subsidize and distribute the CCS
4. Develop a do-it-yourself instructional kit for farmers to build their own CCS

Of the four models selected, we eliminated two because results from research showed that the most feasible and attractive business model to pursue was to start an independent venture or become part of another companys product line. Following a government program wasn't feasible given the little interest of the government in working with privately held farms. Additionally the DIY kit wasnt feasible because we wouldnt be reaching our intended target market and have a positive impact in the Food Insecurity issue in Massachusetts. .

Thus, our teams main decision was whether to start our own company or to provide our product to an already established brand and be part of their product line; to help us make this decision, we developed a pros and cons analysis and voted on the two options based on our available resources, feasibility, and attractiveness (Appendix F). If we were to partner with a company, it would be one of two companies: Grove and Apitronics. Joining their product line would help us get started since we would become part of the already existing distribution channel and established customers. We were also assessing the amount of resources that is needed to start a company, and with further investigation we decided that when this project launches off the ADE pipeline it should do so as its own company. The amount of monetary resources to establish a legal entity in the US and make our first sales is attainable with the funding provided by ADE. Furthermore, we can establish our own first clients with all the contacts we have made throughout the last semester (which neither Grove nor Apitronics have), and we would maximize our profit margin. Our overall risk decreases by starting our own company, as we dont incur the risk of our IP being used by another company or another company simply finding a way to produce their own CCS and taking our CCS out of their product line. Most importantly, we would make sure of having a positive impact in the food insecurity issue in Massachusetts since we would be owning the business and will not have to adapt to the values of another company.

The chosen model a startup called SproutWave will consist of selling a standard version of our CCS (which includes one sensors and capacity to control five outlets) both online and through retail stores. The CCS would contain a set of instructions of how to install it and expand it according to the size of the greenhouse or indoor garden. Furthermore, our model would provide technical support to customers for software updates, product replacement for malfunctioning items, and installation support (Appendix F). In order to understand the whole scope of the business model, we developed a business model canvas (Appendix F) and we have a breakdown of the inventory flow:

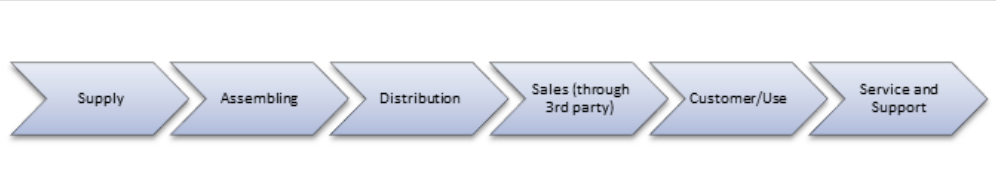


Figure 7.1: Inventory flow of our selected business model - SproutWave.

We would be obtaining the source parts and materials from third parties. The components to build the CCS are as follows:
Receiver Transmitter Computer 2 Sensors (1 air sensor and 1 dirt sensor) 5 Outlets Packaging Instructions Set

The cost of buying the source components to build one CCS off the shelf is approximately $126.00 (Appendix F). However, by obtaining the components for 30 items right from the suppliers, the cost is decreased to about $84.00. All these items would have to be transported to our ADE installations for assembly. The component-manufacturing companies would ship the components to our assembly facility. It would cost our team approximately $4.00 to assemble each CCS and put it in the necessary packaging. Once packaged, the product is shipped to our retailers warehouse in the US. This node might possess some difficulties in it; since we are attempting to place a lot of orders, once we have a steady number of customers,

the abundance of paperwork needed for the certification and tariff processes might oblige us to work with a brokerage firm which would keep track of our paperwork. The challenging part about this is the fact that we would need to sacrifice a portion of our revenue to this brokerage firm.

The advertisement our company would need to place on different media once it is being sold. Especially in the initial stages of our operation, our company will not have a high level of brand recognition since we are starting our own business. In order to obtain a strong position in the industry, as well as magnify our market share, one of the most significant industry nodes we need to focus on is the advertisement node.

Once a CCS reaches our customer, we would need to provide our system with software updates in three month intervals in order to make sure everything is functioning correctly. We will able to remotely monitor and update our systems as long as the CCS is connected to the internet.

7.2.1 Revenue Model

For the initial stage of the business, we have built a revenue model for the indoor farming market which focuses on two streams for earning revenues. The primary stream that will bring in revenue is the unit-sales of the CCS.This revenue model will depend on the number of retailing channels that we find, the price of each product, and the quantity of sales that we may achieve, as shown in the diagram below:

Figure 7.2: Image describing the unit-sales revenue model and its dependents.

An additional revenue stream will be selling the complementary products that you would need to expand the inputs and outputs controlled by our CCS in order to adapt to the needs of larger farms or home growers.

7.2.2 Cost Model

Our cost model consists of four main categories, under which we have analyzed the particular costs. The four main categories are fixed, variable, semi-variable and non-recurring costs that our business model will incur in the future. Our main cost for our product is it the variable cost of our hardware, which can be lowered depending on the quantity that we produce (Appendix F). All of these categories and specific costs were derived from our value chain analysis and can be seen in Appendix F.

7.2.3 Financial Projections

Financial projections were developed to see the feasibility of the project while in the ADE pipeline. We plan to start selling solely in the New England area through greenhouse manufacturers at a price of $150. Given our customer segment studies, there are about 845 small farms in Massachusetts. If we assume that our standard product will reach 10% of those farms in the first year (85) and grows at a 10% growth rate (94

farms in the 2nd year and 103 farms in the 3rd year), then we expect to see the following pro-forma income statement:

	Sept. 2014 - May 2015	Sept. 2015 - May 2016	Sept. 2016 - May 2017
Sales	$12,750	$14,100	$15,450
COGS ($84.00 @ 30 units/order)	$7,560	$10,080	$10,080
Gross Profit	$5,190	$4,020	$5,370
Expenses			
Marketing	$200	$500	$1,000
Administrative	$0	$0	$0
Operational	$1,000	$1,000	$1,000
Other Business Costs	$858	$108	$108
Total Expenses	$2,058	$1,608	$2,108
EBIT	$3,332	$2,412	$3,262
Effective Tax Rate (30%)	$999.60	$723.60	$978.60
Net Income	**$2,132.40**	**$1,688.40**	**$2,283.40**

Figure 7.3: Pro-Forma Income Statement

7.2.4 Long-Term Viability

Given the low cost of production ($84.00), the extensive range of market segments that would use the CCS, the high profit margin of 44% at a price of $150, and the extensive range of geographical regions around the world that could use our device, the project is feasible in the long-run. Looking into more market segments like the personal home-growers and looking at other geographical markets like Europe are only a few of the potential expansion ideas.

7.3 Assumptions and Testing

Our team developed three main basic assumptions to test in order to make sure that we could launch and develop the business. A fourth one surged along the way for further refinement in the next semester.

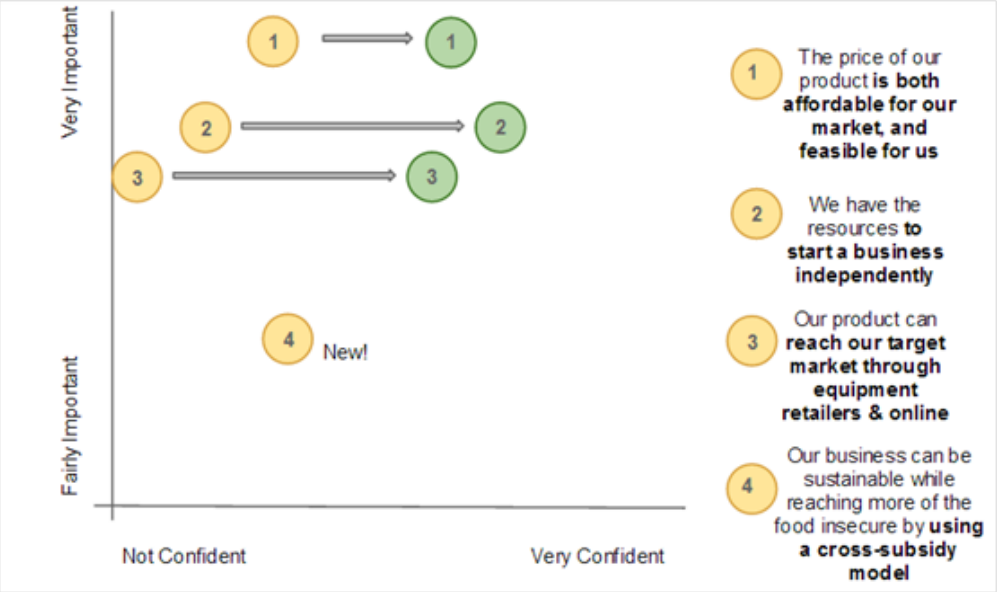


Figure 7.4: Assumptions and progress on each throughout the semester.

Assumption 1: The price of our product is both affordable for our market and feasible for us.
Testing for Assumption 1: We sent buying requests for quantities of 30 and 500 units to each of our potential suppliers for cost analysis. We interviewed local farmers through phone calls and visits and we built five prototypes for assemblage testing and feedback gathering.
Findings for Assumption 1: From our suppliers responses, we found out that the average price would be lower than $84.00 for 30 unit bulks, and $65.00 for 500 unit bulks (Appendix F). Furthermore, we found that small to medium farms would pay $75.00 - $200.00 for the standard version of the CCS and that the willingness to pay increases with the size of the farm. We developed the pro-forma income statement to estimate the profitability of the company while in the ADE pipeline, and it showed a profitable and feasible business in the three-year term with $2,134.40 of net income for the first year, $1,688.40 for the second, and $2,283.40 for the third.

Assumption 2: We have the financial resources to start our business independently.
Testing for Assumption 2: For this assumption, we conducted research on startup and operational costs and developed a breakeven analysis to see if the business will be sustainable solely with ADE funding.
Findings for Assumption 2:
Start-up Costs:
Fixed:
Establish Legal Entity: $750.00
Website & Domain (1 Year): $108.00
Total: $858.00

Variable:
First Bulk (30 units): $2,520.00
Total: $2,520.00

Grand Total: $3,378.00

Breakeven:
The breakeven shows how many units need to be sold in order make back the money we spent in start-up costs.

Fixed Costs/Contribution Margin
Contribution Margin:
Price - Variable Costs
$858.00/($150.00-$84.00)

13 Units; we need to sell 13 units at the price of $150.00 in order to breakeven with our start-up costs. This leaves 17 more units of the 30 unit bulk on the shelf to be sold. If we sell those 17 units at $150.00, we would obtain $2,550.00 which is enough to order our second bulk and keep the business sustainable.

Assumption 3: In regards to distribution, we assumed that we could sell our CCS both through greenhouse equipment retailers and online retailers. Testing for Assumption 3: In order to test this assumption, we visited Griffin Greenhouse Supplies and the hydroponics store and interviewed the store manager to see if they would be interested in selling our CCS. Furthermore, in order to test the online distribution channel, we did some online research on comparable products that were being sold online. Findings for Assumption 3: From our visits to stores, we found that the retailers are willing and able to sell our product and that they sell similar products for prices ranging from about $700.00 to $800.00. From our online research, we found a comparable product named CoolBot that provides cool storage for crops. It sells at the similar price-point of $299.99, has a 1-year warranty, sells solely online, and the company provides tech support; since 2007, they have sold over 14,000 units. Thus, online retailing should be further explored as a potential distribution channel, given the success of this similar product.

7.4 Refinement

Given the wide range of prices that the local farmers are willing and able to pay ($75.00 - $200.00), it would be potentially feasible to pursue a crosssubsidy business model. Following a cross-subsidy model would potentially allow lower income farmers to afford the system and have a larger impact in the food insecurity issue in Massachusetts.

Furthermore, we know that it is financially feasible to keep with the chosen business model while in the ADE pipeline, given the financial projection; however, a post-pipeline financial projection needs to be developed. After launching off the ADE pipeline, the project has to sustain itself, and the team needs the financial projections and analysis to know that the project will be feasible and sustainable over time.

In our business model, there are several components that need refinement and further development to assure that operations can run smoothly once the business is launched. In regards to assemblage, we are assuming that ADE would be able to take care of the assemblage. This needs further refinement because once the business launches off the ADE pipeline, the assemblage will need employees, which will increase the costs of the product.

Now, on the customer and user side, we need to make sure that our team isnt needed for the installation of our CCS by testing if farmers are able to take care of the installation process by themselves. In order

to test this, we would be giving prototypes to the farmers and asking them to install the CCS and give us feedback on the process.

On our end, we also need to refine the maintenance component; once every CCS is installed, we need to make sure we are able to supply farmers with a new component or product if theirs is faulty. Furthermore, we need to make sure that we are able to update our software periodically to ensure it is running effectively. In order to test this, we need to launch more prototypes in the field, gather data, and develop software updates to know if we are capable of doing the maintenance.

Through our research on farms in Massachusetts, we have learned that the majority are established farms that sell high-quality goods to restaurants and CSAs and therefore do not reach the food insecure.

Our studies show the following breakdown of farm types:

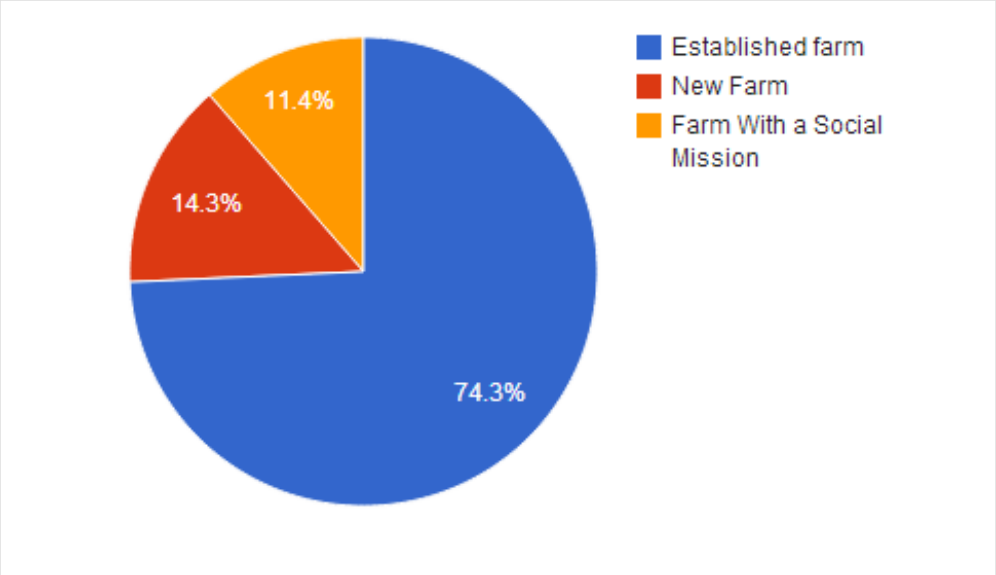


Figure 7.5: Pie-graph showing types of farms and segment in the Massachusetts area. In order to increase food production and make food available to the food insecure, we will follow a business model where we identify and seek partnerships with farms that are connected to food-insecure people. This will be an area of focus for next semesters team.

7.5 Conclusions

We have now have proof of the feasibility of the project, given the SproutWave business model. We are certain that the business would remain profitable while in the ADE pipeline and that our product can reach our intended target market at the price point of $150.00, through which we have a 44% profit margin given the cost of goods sold of $84.00/unit in 30 unit bulks. However, we know now that the cost of goods sold could be decreased down to $63.00/unit in 500 unit bulks, increasing our profit margin. We are certain that we can reach our target market through local retailing shops and are confident that an online distribution channel would also be feasible to pursue given the comparable found. With the aim of lowering the cost for farmers of producing food, we believe that our product can increase accessibility to food of people with lower incomes. In the next few months, we will continue to study the work patterns, needs, and values of

farmers. We are confident that, with the right partnerships and funding, we can accelerate the development and distribution of this product and launch our own company.

8 Social Benefit Analysis

Massachusetts, USA, May 8th, 2014
Anne-Marie Buchenan

8.1 Introduction

Our goal is to address food insecurity by helping increase local food production. We hope to achieve this by helping small farms to propagate seedlings more effectively through the use of our product, a robust, low-cost climate control system (CCS) that decreases seedling mortality and extends the growing season. We are working with several small farms in Massachusetts, most closely with Serving Ourselves Farm (SOS) on Long Island in Boston Harbor.

This report will discuss, in detail, a social benefit analysis (SBA) of this product in the context of SOS, the end result of which will tell us the impact our product can have on the farm. The component started out this semester with a framework in place and some rough numbers calculated; however, when this framework was created, the product we were developing was not defined. Since then, we have more clearly defined our product and, thus, have been able to work with more accurate assumptions and numbers. However, in defining our product, we had to discard much of the work we had previously done. The inputs and outcomes have since been redefined and calculations have been done to monetize these elements. The deadweight, or the scenario without our technology, has been accounted for and the outcomes have been divided by the inputs to obtain the SROI ratio. One of the most important next steps of this component is continuing to discuss the elements of these inputs and outcomes with our partner, SOS, in order to update and validate the numbers used in our calculations. We should conduct a similar analysis for several other farms, to ensure that our product will bring widespread value and that the case of SOS is not unusual or isolated. Lastly, we should average the numbers put into this framework across several farms to find a typical SROI ratio, which we can use to find our total social impact by multiplying by the total number of farms in our market segment.

8.2 Description

Conducting an social benefit analysis of a product-venture can be extremely valuable in aiding development. SBAs determine the impact a venture can have on society by monetizing the inputs (what goes into the system) and the outcomes (the long-term effects of the outputs). Subtracting out what would have happened even without the use of our product and dividing the outcomes by the inputs gives an SROI ratio, or the amount of added social value gotten out of the system with the product in place for every unit of value going into the system. This ratio quantifies how effective a given venture will be in impacting the system. This

information of the impact the venture will have, as well as information regarding the inputs and outcomes, can help determine the most important aspects of a venture and how it can be adapted to provide even more value than originally calculated. In short, an SBA is an invaluable tool in making design decisions.

The first step in analyzing the social benefit of a product-venture is to determine the stakeholders involved and their interests. There are three main stakeholders involved in any farm: the farm manager, the farm workers, and the consumers.

We completed calculations for SOS and these stakeholders became Catalina, those enrolled in the job skills training program, and Bostons homeless specifically. Catalina is concerned with how much money is invested in the farm by its source of funding, the Boston Public Health Commission (BPHC), on materials, equipment, and pay. She is also interested in how much good the allocated money accomplishes. The farm workers involved in the job training program at SOS are interested in the amount of time and effort they put into the farm and the amount of job experience and resulting increased employability they gain. These workers, as well as Bostons homeless people, those being fed by the SOS program, are also interested in getting healthy and nutritious food as well as the results of good nutrition, such as the ability to work harder and earn more.

Given these stakeholders and their interests, a chart can be compiled indicating the various inputs and outcomes that correspond to each group. This chart can be seen in Fig.8.1. The main inputs to the system are labor from the farm workers for planting and growing, as well as money (for equipment and materials) from the BPHC. The major outcomes of the program are an increase in employability of the workers and improved health of Bostons homeless, as well as of the farm workers.

	BPHC	Farm Workers	Boston's Homeless
Inputs	Money	Planting labor; Growing labor	_____
Outcomes	_____	Employability; Health	Health

Figure 8.1: Table listing inputs, outputs, and outcomes of the various stakeholders.

Given these inputs and outcomes, we can see how adding a CCS will have an effect. The money put into the program goes into materials and equipment. For materials cost, the focus is mainly on seeds, trays, and soil for propagation. Using amounts and prices of the farms most commonly used seeds, trays, and soil allowed the total materials cost to be calculated. These numbers were assumed to decrease with the use of our product, because our product will reduce seedling death, meaning fewer materials are needed to get the same number of plants. For equipment costs, we considered only the added cost of our product, since all other costs are assumed to be constant with and without the product. For labor inputs, we based the calculations on the wages of the workers. The labor value for planting was assumed to decrease with our product, since less seedling death means fewer seeds required planting. The growing labor value was also assumed to decrease, as the CCS would take over many responsibilities of the workers. The values for each input without and with our product can be seen in Appendix G.

Outcomes of the SOS program are increased employability for trainees and better health for homeless people. Employability was monetized using the difference between the percentages of homeless people who

get jobs with and without the program, which was multiplied by the amount of money they would make per year at that job. The value of employability will increase, because our product will free Catalinas time enough that she can invest extra time training more workers. Improved health numbers were based on the cost of healthcare per homeless person based on average amount of time spent in the hospital; this is expected to remain unchanged since we are assuming the product will not increase the amount of food grown, meaning same number of people are being fed by the program. The total values of each outcome with and without the product are displayed in Appendix G.

The calculations for all inputs and outcomes can be seen in Appendix G. The total values for these can be extracted from the tables in Appendix G. Then the sum of the outcomes without the product is subtracted from the sum of them with the product to get the total value added by the product, accounting for what would have happened without our product; in other words, this results in our impact. Lastly, the difference of outcome value is divided by the total input value to the system given our product to get the SROI ratio. We found a total increase in value of outcomes of $9,634 and value of inputs of $1,867 with our product. This results in an SROI ratio of 5.2 to 1, which is reasonably high. A summary of all of the inputs, outcomes, and the SROI ratio is shown in Fig.8.2.

	Inputs	Outcomes	SROI Ratio
Without Product	-$2,881	-$5,239,324	
With Product	-$1,867	-$5,229,690	
Impact		*$9,634*	***5.2 : 1***

Figure 8.2: Table of total input and outcome values shown with and without the product, as well as the calculated SROI ratio.

This ratio is not final, as our project is still in the development stage. Many assumptions were made and many numbers involved in the calculations were estimations based on the believed effect of our product. This component will progress more as the numbers are revisited as we test our prototypes through the upcoming propagation season. However, much of the work is done, as the entire structure of analysis has been defined. It will be an easy matter to update the SROI ratio as more accurate numbers are obtained.

8.3 Assumptions and Testing

In order to monetize the inputs and outcomes, many assumptions had to be made, especially as our product is still in development and its effects are not accurately known. These assumptions are listed in Appendix G.

We made some assumptions regarding our products effect, namely that it would eliminate seedling mortality, that it would reduce labor during growing by 90%, and that it would cost $150. These assumptions will be verified as we test our prototypes and as we solidify a parts list for the CCS with prices.

In order to monetize our inputs, we assumed that SOS will grow 94 trays of seedlings this season. The assumptions regarding these types of plants and their ratios are supported by spreadsheets of the 2013 seasons production, showing the pounds produced of each plant. The assumption about prices is supported

by research available in Appendix G. Next, we assumed that each 60-quart bag of soil costs $29.50 and that the farm uses 36 bags for propagation (Johnny Seeds). These numbers are supported by research and by conversations with Catalina, respectively. We assumed that all material inputs would decrease by 20% with our product, as we learned from Catalina that 20% more seeds than the desired number of plants are planted. For labor, we assumed that workers can plant three trays per hour and that total planting time would decrease 20% with our product as fewer seeds would need to be planted. Next, we assumed that growing seedlings requires 3 hours of work per day for 5 weeks. Lastly, we assumed the workers time is worth minimum wage, which is reasonable as the workers have little job experience.

For our outcomes, we assumed that the homeless have a 2% chance of getting a job without training and a 60% chance with the SOS training. This first number was based on research into the fact that only 2% of homeless have full-time employment (Crisis). The second number was supported by the SOS website which cites that six out of ten workers that go through the program are placed in jobs (Friends of Boston's Homeless). We also assumed that those who do obtain jobs will make minimum wage, which is reasonable as they still have relatively little experience. We assumed that, given our product, Catalina would have more time to train another worker. Additionally, we assumed that it costs $2,341 per day in the hospital for an average person in MA (Becker's Hospital Review), that the average person stays in a hospital for 4.8 days (Centers for Disease Control and Prevention), that a homeless person spends an average of 4 days more per visit in the hospital than the average person, and that there is 32.2% chance of a person staying in the hospital in any given year (National Alliance to End Homelessness). All of these numbers are backed by research. These healthcare numbers are assumed to remain the same with and without our product.

As we can see, this SBA requires a variety of assumptions, most of which have been validated through fifty-cent tests involving research and conversations with Catalina, the SOS farm manager, who can be reached at clospina@bphc.org. The assumptions regarding the products effects will be refined by $500 tests involving building and testing prototypes during the propagation season.

8.4 Refinement

So far, we have made significant progress on this component this semester. We began with a framework and rough numbers and, through the semesters course, reevaluated the framework and entered more accurate numbers as we better defined our products effects. Despite working with a more well-defined product, this analysis still required numerous assumptions. Most assumptions were backed by extensive research, and the remainder were supported by information directly from Catalina.

The new framework is in place, but the numbers require further refinement to increase the analysiss validity. In the future, we can refine these numbers by updating our assumptions about the products effects, as more information becomes available through our prototype testing. Additionally, these numbers can be refined by reevaluating the involved stakeholders, inputs, and outcomes. Another stakeholder group to consider is the food insecure buying from SOS at farmers markets. The value that the BPHC receives out of the system may be important to identify. Perhaps the committee is able to look better publicly. Beyond this, the increase in quality of life due to the decrease of stress for the farmers should be quantified, as well as the value that those farmers can add through their newly freed work hours. What else might they be able to accomplish now that the CCS has taken over many of their responsibilities?

In addition to refining the numbers for a SBA of SOS with our product, future work could include applying the analysis to SOS alone, considering what the outcomes would be without the farm program; this

analysis would be quite valuable to Catalina. Additionally, the analysis could be applied to other farms to give them an idea of the impact of our product. Additionally, we can apply this framework to a typical farm and multiply by the number of farms in our market segment to get an idea of the total benefit our product can achieve.

8.5 Conclusions

Our SROI ratio for SOS is 5.2 to 1, which is a promising figure. This means that every dollar invested in SOS, when they are using our product, will return a social value that is $5.20 more than the value that would have been returned without our product. This number will be updated as information is gathered from the field test of our prototypes. This general SBA framework will be applied to other farms. Additionally, it will be applied to typical farms and multiplied by the number of farms in our market segment to find our overall benefit. All of these refinements will help us get a better sense of how much of an impact our product will have and make more informed decisions regarding development.

9 References

1. "2014 Water and Sewer Rates." Boston Water and Sewage Commission. Boston Water and Sewage Commission, n.d. Web. 07 Apr. 2014.

   This source lists the prices of water per 1000 gallons in Massachusetts.

2. "Average Cost Per Inpatient Day Across 50 States in 2010." Becker's Hospital Review. Becker's Hospital Review, n.d. Web. 07 Apr. 2014.

   This source lists the average cost per person per day in the hospital in various states, including Massachusetts.

3. "Browse Our Organic Selections." Organic Seeds, Plants, & Supplies. Johnny Seeds, n.d. Web. 07 Apr. 2014.

   This site lists prices of seed packets for various plants, which can be used to find the unit price per seed. It also lists prices for propagation trays.

4. "Cost of Homelessness." National Alliance to End Homelessness:. National Alliance to End Homelessness:, n.d. Web. 07 Apr. 2014.

   This gives the average extra time spent on in the hospital per visit by the homeless.

5. Greenhouse Megastore. Greenhouse Megastore, n.d. Web. 07 Apr. 2014.

   This site gives prices for a variety of propagation needs.

6. "Hospital Utilization." Centers for Disease Control and Prevention. Centers for Disease Control and Prevention, 28 Feb. 2014. Web. 07 Apr. 2014.

   This gives the average length of stay per visit in the hospital by the average person.

7. "The Farm @ Long Island — Friends of Boston's Homeless." Friends of Boston's Homeless. Friends of Boston's Homeless, n.d. Web. 07 Apr. 2014.

   This is the website for SOS Farm. It provides information on how much food they produce and how many homeless they serve. It also has a figure telling us how many program graduates are placed in jobs, allowing us to calculate the increase in employability.

8. "Work and Skills." Crisis. Crisis, n.d. Web. 11 Apr. 2014.

   This source gives us numbers for how many homeless people are employed.

9. "U.S. Department of Agriculture." U.S. Department of Agriculture. U.S. Department of Agriculture, n.d. Web. 07 Apr. 2014.

The website for the USDA provides a wide variety of background information pertaining to agriculture, such as facts pertaining to food and nutrition. Numbers based on the 2013 seasons production come from the Harvest_2013 spreadsheet given to us by Catalina and found in ADE Massachusetts / 3 Fall 2013 / Numbers/Data.

Appendix A

Information for Project Manager in Fall of 2014

This appendix will talk about some of the larger system-wide assumptions and choices we've made, farms we've become partnered with, and potential showstoppers as well as providing you with a short guide to how our information, our team, and our class is organized.

System-Wide Assumptions and Choices

There are a few assumptions and choices we've made that have a prominent system-wide effect. We've decided that our product is a climate control system which uses parts purchased cheaply off the shelf and some very smart code to automate the growing process. This means we have a product that relies heavily on electrical and computer engineering, and much less on mechanical engineering, unlike other ADE projects. We have also decided that this product will eventually leave the ADE pipeline in the form of a startup, as opposed to being part of an existing company's product line or as an open-source, instructables-style, free-for-everybody set of instructions. Our final major assumption, which every semester's team grapples with, is whether or not we should assume that increasing the output of local farms will actually help to reduce food insecurity. We have chosen to believe this assumption for the following reasons:

- Leading experts say so. We don't have time to do the research, but other people do, and have found it to be true. We don't, however, have official citations for those findings, so that might be a good thing to find to validate this direction. Asking Kofi is a good way to go about this research.
- Many farms that we have spoken to accept food stamps as payment for their produce. SNAP, the food stamp program, is a great way to track how much we are reaching the food insecure.
- No one product or service is going to "solve" food insecurity. We are not looking for the perfect answer, we are just making one solution among many to reduce this symptom of a much larger problem.

Contacts

Much of our work over the past two semesters has been to show that there is a market out there of small, local farms that can serve the food insecure. We have picked three organizations in particular to work with, Serving Ourselves Farm (SOS), New Entry Sustainable Farming Project (New Entry), and Nuestras Raices (NR).

SOS is located on Long Island in Boston Harbor. They are associated with a homeless shelter and soup kitchen which are also on the island. A significant portion of their produce is donated to this kitchen to feed the homeless in the shelter. Our contacts at SOS are Catalina Lopez Ospina (the farm manager) and Belkis Roman (SOS' contact with the Boston Public Health Commission, which funds the farm). They are both amazing, friendly, helpful, and extremely busy all the time. They have no annual funding for improvement projects and propagate their seedlings in Catalina's office and in a hallway in the abandoned chapel on the island because those are the only places with electricity that they have access to.

New Entry is an educational organization that offers classes as well as farmer training programs with designated plots of land for learning farmers. They have a huge network of new farmers and farmer education programs all over the US. Our contacts there are Eva Agudelo Winther, National Incubator Farm Training Initiative (NIFTI) Coordinator; Sam Anderson, Livestock Program and Outreach Coordinator (who does the initial workshop); and Eero Ruuttila, the farm manager, who never responds to email. More information can be found on their website, nesfp.org.

Nuestras Raices (Our Roots) is a new partner that we found this semester at the Urban Farming Conference at Northeastern University. They are a "grassroots organization that promotes human, economic, and community development in Western Massachusetts, through projects relating to food, agriculture and the environment." We spoke with Anne Cody, Director of Operations and Planning, and she was very excited to implement a prototype at NR as soon as possible. We were unable to put a prototype at their farm this semester due to the sudden influx of work that is spring on a farm, but they are excited to work with us in September.

Contact information for these farms and many others can be found in the Farms and Contact Record document in the Site Visits folder.

Showstoppers

"Showstoppers" are things that can prevent a team, product, or venture from getting off the ground. Consider this your crystal ball of things-that-will-go-wrong.

On a logistical level, a lack of communication can ruin a project. With teams as large as these, it takes a constant and deliberate effort to keep everyone apprised. ADE requires one out-of-class meeting per week. You should schedule this very early on, and invite Kofi Taha, the team MA advisor, to skype in to each one of those. You should also schedule once-weekly (at least) sub-team meetings, however you decide to define those sub-teams. You should try to get everyone to go on site visits, but it will be impossible and inconvenient to send everyone on every site visit. Having more than five people on a site visit gets hectic. Schedule visits according to when the farms are available, as long as at least 3 people can go, and then have people sign up for those dates rather than providing the farm with a list of preferred days/times. It is good to know, though, when people are available on an average week. For example, this semester, if there is an event on a Wednesday or Friday afternoon it's a good bet that at least 3 people will be able to go.

The areas of concern in the nuts and bolts of the project are:

- Product
    - Communication. We were using the Centerpoint from Verizon (graciously donated), but it's expensive and isn't robust. We need to communicate with the system for debugging and software updates now, but in the future we'll need it internet accessible so that it can utilize a web-based UI. We have since replaced it with a hand-held radio, but this system still needs polishing.
- Business
    - Pricepoint. Finding the butter zone where the product is being sold for enough to make the project/business economically sustainable while still being affordable for our target market.
    - Marketing. Our target market has interesting marketing inaccessibility issues, and tapping into existing networks will be essential to selling product. We now have the fundamentals of marketing materials, a design guide and logo, available in the branding folder in the SP'14 google drive folder.
    - Moving past ADE. When our product is ready, this project will move out of the ADE pipeline. How and when it does that deserves considerable thought. We've decided that it should take off as it's own startup, but that's easier said than done.
- User
    - Designing for Market Segments. Because our system is so versatile, there is a wide variety of market segments to which it could be marketed. However, designing for an immigrant farmer who speaks minimal english and wants to grow crops from her homeland in the ¼ acre she's leasing at a community farm is very different than designing for an industrial bulk seedling distributor. Deciding what market segment(s) to design for, and then designing specifically for them will be very difficult.

    - Accommodating Needs. On a related note, many of our potential customers have extremely important and specific needs. Everything from language-barrier problems to analytics to remote crisis management. Artfully solving these problems will be a marathon, not a sprint.
- Integration
    - Right now, we have a lot of interesting user research and an increasingly functional robust technology. Integrating our user design into the product itself will be non-trivial. This transition from "technology" to "product" involves everything from branding, to part sourcing, to user design, and it would not be amiss to have one person's entire semester devoted to this.

Organization

There are three different types of organization that would be helpful for the future PM of ADE-MA to understand: the information, the team, and the class.

The information is all in the Google drive folders. Please do not start up a dropbox or network drive folder. We also have a large amount of code stored on Github, but everything that isn't Solidworks or Software should be in the drive. I've already provided a suggested layout for your file structure and some instructions as how to use the folders in the drive. This is not mandated by the class, just some wisdom passed down. The system described therein was developed through two years of trial and error, and the less time you spend fiddling about with file structures, the more time you spend focusing on the project. The most important document in there (other than these component reports) is the Farms and Contact Record that can be found in the Site Visits folder. We started this document this semester to keep track of potential partner farms and if/how we had contacted them. I recommend changing it to suit your styles and needs, but having that information readily available is extremely important.

All externally facing information can be found on the blog, ade-ma.tumblr.com (Username/email: ade-ma@googlegroups.com, Password: TheCoolest), and our website, sproutwave.com. Sproutwave.com was purchased by Ian Daniher and is currently hosted on mylanderpages.com, as designed by Colby Sato, but we are re-working it. Creating a clear and consistent external image (website, branding, business cards, informational flyers) would be a great job for next semester's Value Proposition team. Design guides, logos, and icons can be found in the SproutWave Branding folder in the google drive.

The organization of the team can be a major point of contention and failure in any project, especially ADE where there is an implied "norm" for how teams are split up (by component) and there is a mix of students from Babson, Olin, and Wellesley. There is no right answer, for the class or for the team. This is the sort of thing you have to be ready to change if it isn't conducive to your team's productivity. This semester, we ended up splitting into "TUB" (Technical, User, and Business) teams, rather than the standard component sections. The Technical team worked on the Product/Service System and Comparables and Prior Art. The User team worked on Customer Needs and Market Segmentation and the Social Benefit Analysis. The Business team worked on the Business Model and Value Proposition. This worked well for us. It meant we weren't spread thin, were able to devote more people to each subteam, and had an easier time scheduling meetings. This also encouraged more intra-team communication, which is always a good thing. Unfortunately, it meant that components like Social Benefit Analysis and Value Proposition were not pursued as doggedly.

ADE is an experiment, and it's still in its baby years, which means that occasionally the students in the class are left a bit adrift while the professors figure out what's going on and how to react. To help you out, I've prepared a few helpful documents. In the outermost ADE folder there are three documents: a template for these Component Reports, a presentation that summarizes the CPR rubric and what your CPR should have, and a presentation titled "HOW TO WIN AT

ADE," which is fairly self-explanatory. The entire system is in flux right now. If some changes Kat Brookshire and I are suggesting are adopted, most of these documents will be rendered obsolete, but that is a good thing.

Next Steps

ADE starts fast and never slows down, so here's a short list of some things you should do in the first few weeks to make sure you make the most of your semester.

- Step one is scheduling your out-of-class meeting time. It's worth the time to do this during the first class. The sooner you start having those meetings, the better. And the sooner you begin trying to schedule them, the more likely you'll be able to find a suitable time.
- Start making meeting notes from the very beginning. More tips and tricks about this can be found in the "How to use this folder: meeting notes" document in the meeting notes folder, but the sooner you start documenting your work, the better the communication within your team will be.
- ADE will ask you to do a learning styles assignment. Do this as early as possible and use that discussion to decide who will be taking on what roles and what components people would like to own. This is a good time to figure out how you're going to split up the team.
- Get some space in the Olin classroom for your team to use. Talk to the NINJA, Ben Linder, and any other professors that are teaching in the room and ask for some wall space for frameworks and enough floor space for the rolling shelves/prototype. Having dedicated space is important for getting work done efficiently and bolstering a sense of ownership in the class.
- Make contact with SOS, Nuestras Raices, and New Entry as soon as you can. Politely introduce yourself as the new project manager of ADE-MA and ask when you can set up visits.
- Update the Farms and Contact Record document, and make it into a form that will be useful to you in the coming semester.
- Take the time to talk about the long-term goals of the project and lay out where you want to be by the end of the semester. Lay out internal deadlines and deliverables to get you there on time. Pick which components you'll be presenting at which CPRs. Post your timeline and goals somewhere prominent so as to not lose sight of them as the semester gets crazy.
- Send out a survey to see when people are generally free for the rest of the semester. This will be vital information when you start to hear back from the farms as to when you can come and visit.
- Visits will be useful all semester long; schedule as many of them as you can manage without overlapping awkwardly.
- Finally, take the time to set up the classroom prototype and start growing plants. Not only is it a good test and a morale-builder, but also an excellent source of tasty vegetables.

Each semester seems unbelievably short and we get less done than we hope to. This is a list of long-term goals and tasks which might serve as a jumping-off point for the team.

- A web interface for the CCS
    - Having one at all would be nice
    - Long term, making it simple, elegant, informative, and meeting our users needs (multi-lingual, easily visible, etc…)
- Streamlined setup procedure
    - Have a list/kit with every part you need to set up the system, documented, and readily available. You never know when a farm might ask you to install your prototype in two days time.

    - Long term, have the system capable of identifying what outlets affect the environment in what ways without being told what they're connected to. For example, instead of telling the computer that outlet #4 is connected to fan, it should realize that when outlet #4 turns on it gets cooler and dryer and use that information to control the climate.
- Actionable user insights
    - Turn our user research into design decisions to improve the project.
- A clear brand and external presence
    - This semester we chose to call our system "SproutWave"
    - We also had a first draft of a website, and developed a logo and design guide.
    - This coming semester we need a more cohesive and well designed media presence. Everything from an informational landing page to business cards to pamphlets we can hand out at farmers markets or on site visits.

Appendix B

Customer Needs

This is important enough to note by itself, so I am noting it here in relation to picking user visits: There is a list of farmers markets in Massachusetts that proved invaluable in helping us find markets to visit; http://www.massfarmersmarkets.org/. If you mouse over the Find button on its main page, you can click markets, which will take you to a list of markets. An important thing about this list is that it says if the market accepts EBT/WIC/foodstamps/boston bounty bucks, i.e. the markets that cater to the food insecure.

The images for the personas were too large to fit in this document, so here there are on the web:
First persona http://imgur.com/DbtPR7V
Second persona http://imgur.com/GyMjC1o
Third persona http://imgur.com/YxzFqY6

Appendix C

Market Segmentation

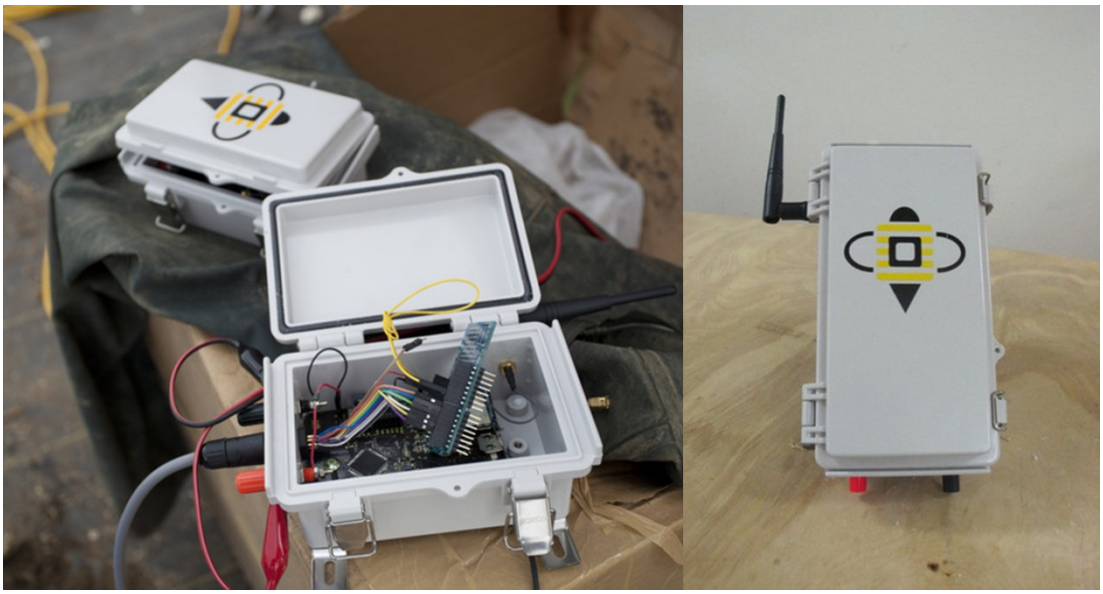
Sources:

1. USDA 2007 Census of Agriculture, Massachusetts State and County Data
    a. Extensive list of tables and reports about farms in Massachusetts. Data should be taken with a grain of salt because their definition of "Farm" is quite broader than ours.
2. Beginning Farmers Network. Farmer Profiles. http://bfnmass.org/profiles/
    a. Persona-like profiles, compiled by Bentley University students on local farms and farmers of all types, as well as people working to help the small-farm community.
3. Keough, Gary. 2012 Census Results on Beginning Farmers. http://bfnmass.org/blog/2012-census-results-beginning-farmers.
    a. This is a useful synopsis of the latest agricultural census data in Massachusetts, especially looking at the numbers for new farms and minority farmers.
4. New England Small Farm Institute. About New Farmers. http://www.smallfarm.org/main/for_service_providers/about_new_farmers/
    a. Quick set of guidelines about the needs and backgrounds of new farmers in New England.
5. New Entry Sustainable Farming Project. Our Farmers. http://nesfp.org/farmers
    a. Persona-like profiles of real farmers working with New Entry.
6. Severine vT Fleming. "The Greenhorns." Documentary 2010 http://www.amazon.com/The-Greenhorns-Severine-Tscharner-Fleming/dp/B00BFXYY4E
    a. Informative documentary with excellent interviews and quotes from young farmers in the US. It brings out their goals, values, struggles, and achievements.
7. Greenhorns.net. Guidebook For Beginning Farmers. 2010. http://www.thegreenhorns.net/wp-content/files_mf/1335219697greenhorns_guide_sept2010_web.pdf
    a. Illustrated zine that describes techniques, issues, and resources for beginning farmers. Brings out goals and values, especially with sketches and illustrations.

Appendix D

Comparables and Prior Art

"Field-Ready Bee" by Apitronics
http://store.apitronics.com/collections/frontpage/products/field-ready-bee



Price: $240
Functions: Versatile environmental control - compact sensor/control box which can be mounted but still opened for easy interior access
Downsides: Requires a large amount of initial user input to run, as it must be programmed for a specific use before it is used to sense and/or control different aspects of the environment, price
Features Desirable to Our Product: Wireless sensing, wireless relay of climate information for a short distance (up to 9 miles), battery and solar power, waterproof

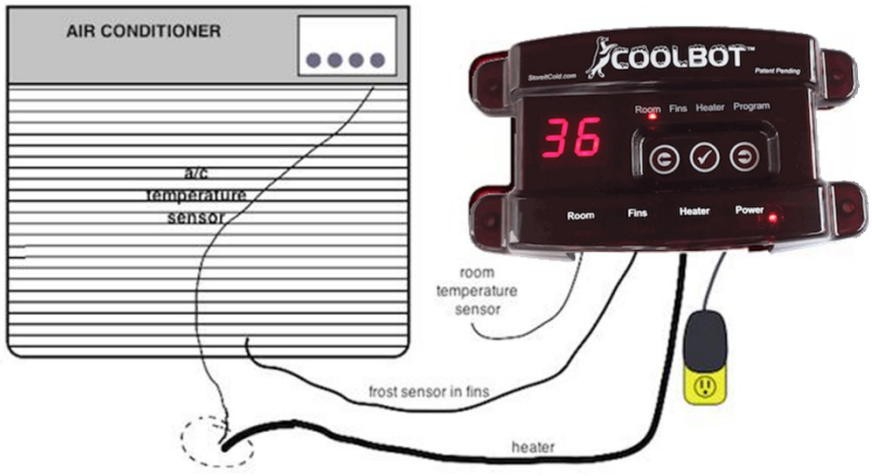
"iGrow" by Link4
http://link4corp.com/products/igrow100.php

Price: $625
Functions: Supports 4 outlets, touch-screen interface, environmental control
Downsides: Limited functionality, price
Features Desirable to Our Product: User-friendly interface, can be monitored or controlled from mobile device, data-logging capability, wireless monitoring, real-time data, can track energy usage of device

“Autopilot Greenhouse Master Controller”

Price:$600
Functions: 4 outlets; sensor bundle: temp, humidity, CO2 (1 each); Optional USB modem
Downsides: Control software does not run on Macs or Smartphones, hard-wired, unintuitive interface, power limited to one outlet, price
Features Desirable to Our Product: Dependability (warranty)

“CoolBot”



Price: $299
Functions: Controls a standard AC unit; Senses frost, room temperature, and AC temperature
Downsides: Only designed for refrigerators - limited functionality, hard-wired, power limited to one outlet, unintuitive interface, price
Features Desirable to Our Product: Successful independent startup, relatively low price, marketing and network

“Grove”

Price: TBA
Functions: Sensors and actuators TBA, cloud connectivity, beautiful interface
Downsides: Hard-wired, price
Features Desirable to Our Product: Connectivity, interface, modularity, web presence

Appendix E

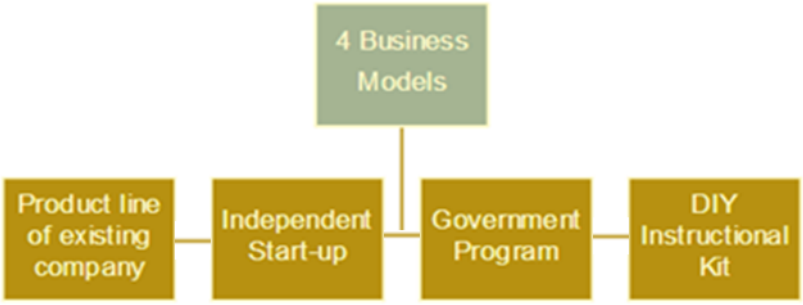
Product/Service System

- tour de github
- ratpak writeup
- process network introduction
- BOM
- aprs101
- ifsk / gold code verizon solution

Appendix F

Business Model

Images from ideation phase and final four business models to pursue. First Image shows all the ideas to bring the product to the market. The second shows the four best potential business models to follow.



Pros and Cons analysis of final two business models

Joining Product Line
Pros:
More funding support
Access to distribution channels
Brand recognition

Cons:
End goals are different
Market segments are different
Little control of prices

Grovelabs

Apitronics

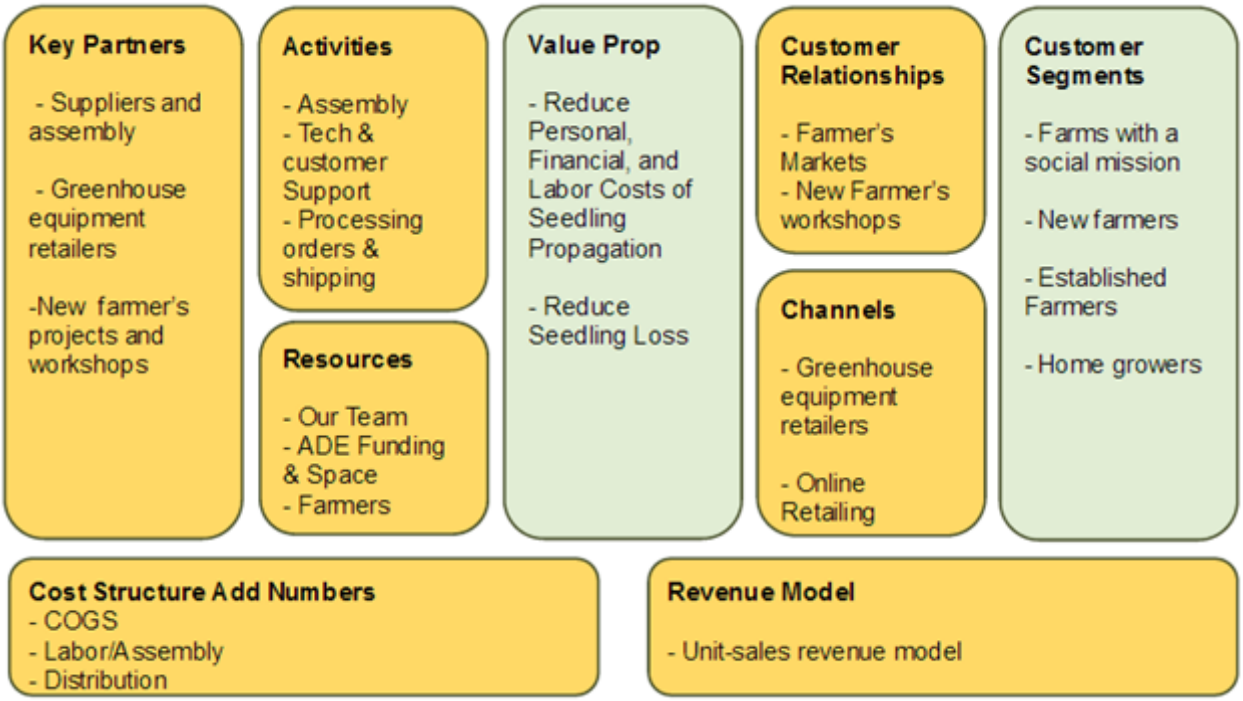
SproutWave

Independent Startup
Pros:
Control of price point and distribution
Can work on our own timeline

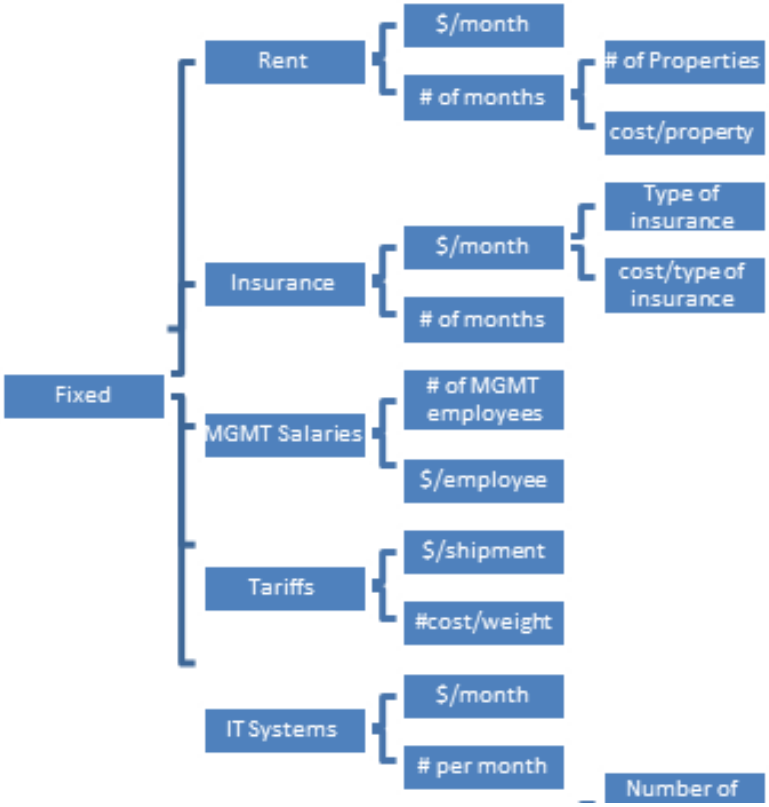
Cons:
Less funding

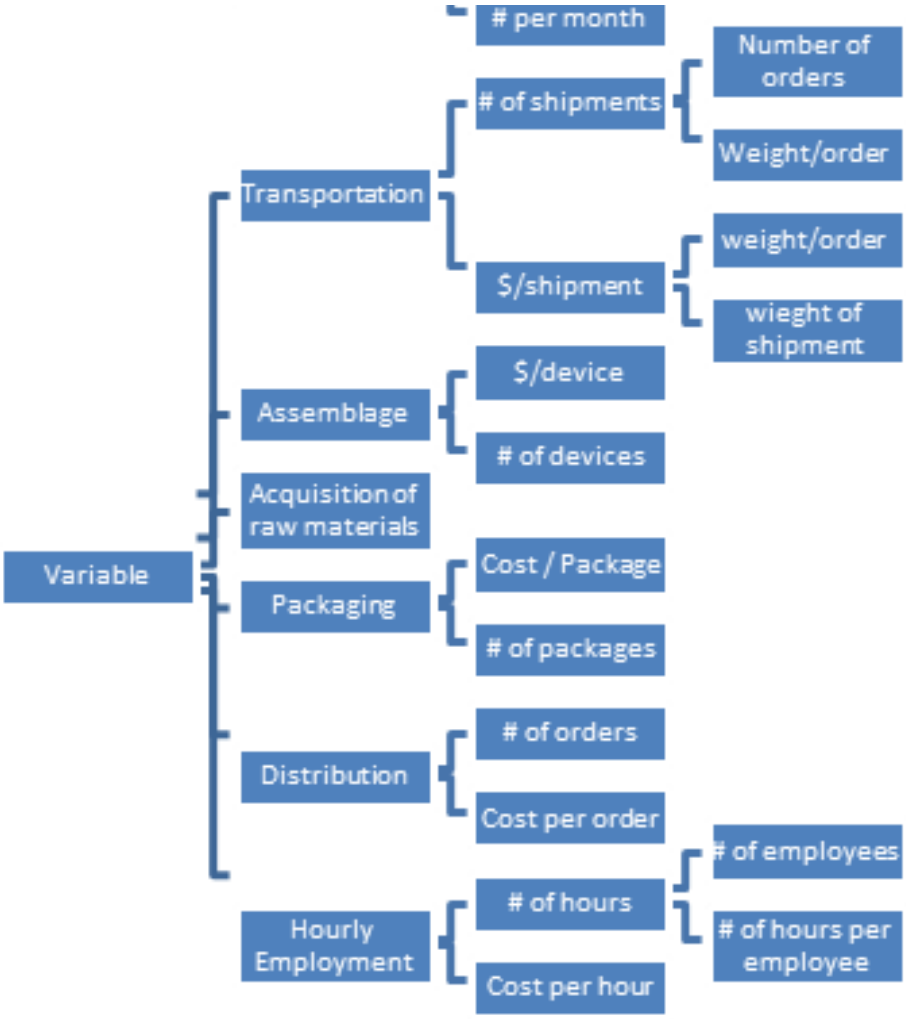
Our business model provides a product, the components, and the service to keep the system functioning for our customers.

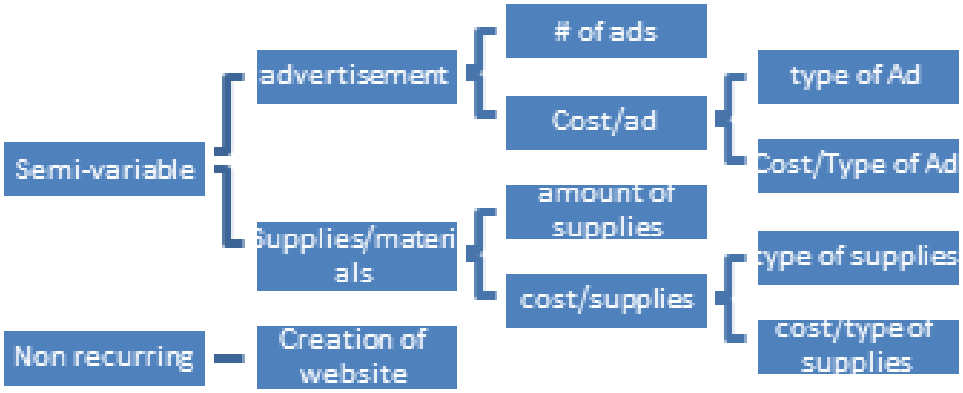
Business Model Canvas for SproutWave:



Cost Model for Business Model




per month
Number of orders
# of shipments
Weight/order
Transportation
weight/order
$/shipment
wieght of shipment
$/device
Assemblage
# of devices
Acquisition of raw materials
Variable
Cost / Package
Packaging
# of packages
# of orders
Distribution
Cost per order
# of employees
# of hours
Hourly Employment
# of hours per employee
Cost per hour


# of ads
advertisement
type of Ad
Cost/ad
Cost/Type of Ad
Semi-variable
amount of supplies
Supplies/materials
type of supplies
cost/supplies
cost/type of supplies
Non recurring
Creation of website

These tree diagrams show the costs that our venture will have to incur once it launches out of the ADE pipeline. The diagram shows the fixed costs or costs that will not change depending on how much we sell, the variable costs or costs that are dependent on how much we sell, the semi-variable costs or costs over which we have control how much we spend, and the non-recurring costs or costs that we will have only one time. Given that we will be selling a product our main incurring cost will be the variable cost of producing and assembling each climate control system. This cost will depend on the demand of the market and how many we produce and assemble.

Cost of Goods Sold per Quantity

Components:	1 Unit	30 Units	500 Units
Receiver	$9.00	$8.00	$6.00
Transmitter	$10.00	$3.00	$2.00
Tiny Computer	$40.00	$30.00	$25.00
Sensors (2)	$27.00	$14.00	$10.00
Outlets (5)	$40.00	$22.00	$16.00
Remote	$0.00	$0.00	$0.00
Manual	$0.00	$0.00	$0.00
Transportation	$0.00	$3.00	$3.00
Packaging	$0.00	$4.00	$1.00
Labor	$0.00	$0.00	$2.00
Cost of Goods Sold	**$126.00**	**$84.00**	**$65.00**

Appendix G

Social Benefit Analysis

Key Value Summaries

Inputs	Without Product	With Product
Seeds	-$467	-$373
Trays	-$261	-$209
Dirt	-$1,062	-$850
Equipment	$0	-$150
Planting labor	-$251	-$201
Growing labor	-$840	-$84
Total	*-$2,881*	*-$1,867*

Outcomes	Without Product	With Product
Better health	-$5,306,766	-$5,306,766
Increase in employability	$67,442	$77,076
Total	*-$5,239,324*	*-$5,229,690*

Calculations

Input Calculations

- Seeds

6026 seeds * ($0.003/bok choy seed[3] * .088 bok choy + $0.002/beet seed[3] * .119 beets + $0.008/cabbage seed[3] * .084 cabbage + $0.002/carrot seed[3] * .153 carrots + $0.002/collard seed[3] * .088 collards + $0.192/cucumber seed[3] * .078 cucumbers + $0.043/eggplant seed[3] * .086 eggplant + $0.004/onion seed[3] * .084 onions + $0.54/potato seed[3] * .096 potatoes + $0.04/summer squash seed[3] * .124 summer squash) = $466.81

6026 seeds * 0.8 * ($0.003/bok choy seed[3] * .088 bok choy + $0.002/beet seed[3] * .119 beets + $0.008/cabbage seed[3] * .084 cabbage + $0.002/carrot seed[3] * .153 carrots + $0.002/collard seed[3] * .088 collards + $0.192/cucumber seed[3] * .078 cucumbers + $0.043/eggplant seed[3] * .086 eggplant + $0.004/onion seed[3] * .084 onions + $0.54/potato seed[3] * .096 potatoes + $0.04/summer squash seed[3] * .124 summer squash) = $373.45

- Trays

$2.59/open tray[3] * 50 trays + $2.99/50-cell tray[3] * 27 trays + $2.99/128-cell tray[3] * 17 trays = $261.06

($2.59/open tray[3] * 50 trays + $2.99/50-cell tray[3] * 27 trays + $2.99/128-cell tray[3] * 17 trays) * 0.8 = $208.85

- Dirt

$29.50/60-quart bag[3] * 36 bags = $1,062
$29.50/60-quart bag[3] * 36 bags * 0.8 = $849.60

- Equipment
  - $0
  - $150

- Planting Labor
  - $8.00/hour * 1 hour/3 trays * 94 trays = $250.67
  - $8.00/hour * 1 hour/3 trays * 94 trays * 0.8 = $200.53

- Growing Labor
  - $8.00/hour * 3 hours/day * 7 days/week * 5 weeks = $840.00
  - $8.00/hour * 3 hours/day * 7 days/week * 5 weeks * 0.1 = $84.00

Outcome Calculations

- Better health
  - $2,341/day[2] * 8.8 days/visit[4] * 0.322 visits/person[4] * 800 people[7] = $5,306,766.08
  - No change.

- Increase in employability
  - $8/hr * 40hr/week * 52 weeks * (0.60^{7} - 0.02^{8}) * 7 workers = $67,441.92
  - $8/hr * 40hr/week * 52 weeks * (0.60^{7} - 0.02^{8}) * 8 workers = $77,076.48

Assumptions

Input Assumptions

- SOS plants 6026 seeds per season, based on 50 open trays of 50 seedlings, 27 trays of 50 seedlings, and 17 trays of 128 seedlings.
- Open trays cost $2.59, 50-cell trays cost $2.99, and 128-cell trays cost $2.99 (Johnny Seeds).
- Bok choy, beets, cabbage, carrots, collards, cucumbers, eggplant, onions, potatoes, and summer squash each make up 8.8%, 11.9%, 8.4%, 15.3%, 8.8%, 7.8%, 8.6%, 8.4%, 9.6%, and 12.4% of plants, respectively.
- Bok choy, beets, cabbage, carrots, collards, cucumbers, eggplant, onions, potatoes, and summer squash seeds cost $0.003, $0.002, $0.008, $0.002, $0.002, $0.192, $0.043, $0.004, $0.54, and $0.04, respectively (Johnny Seeds).
- Each bag of 60 quarts of soil costs $29.50 (Johnny Seeds).
- SOS uses 36 60-quart bags of soil.
- Our product will eliminate plant mortality, meaning that all material costs will decrease by 20%, since SOS currently plants 20% more seeds than the number of plants desired to be transferred to the field.
- Our product costs $150.
- Workers plant 3 trays in 1 hour.
- Workers spend 3 hours per day watching over seedlings for a total of 5 weeks before they are transferred to the field.
- Our product will reduce growing labor by 90%.
- Farm workers' time is worth minimum wage, or $8.00/hr.

Outcome Assumptions

- Homeless people have a 2% chance of getting a job without training (Crisis).

- Homeless people have a 60% chance of getting a job after a 126 day program at SOS (Friends of Boston's Homeless).
- A homeless person that gets a job will make minimum wage.
- SOS will employ 7 workers without our product, but Catalina will have time to train 8 workers as our product frees up her time.
- It costs $2,341 per day in the hospital for an average person in MA (Becker's Hospital Review).
- The average person stays in the hospital 4.8 days per visit (Centers for Disease Control and Prevention).
- The homeless spend an average of 4 more days per visit to the hospital than the average person (National Alliance to End Homelessness).
- There is a 32.2% chance of the average person making a visit to the hospital (National Alliance to End Homelessness).
- The cost of healthcare for the homeless will not change with the use of our product, as the farm will not produce more food and, thus, will not feed more homeless.

Appendix H

Environmental and Sustainable Entrepreneurship
Janna M. Zimmermann

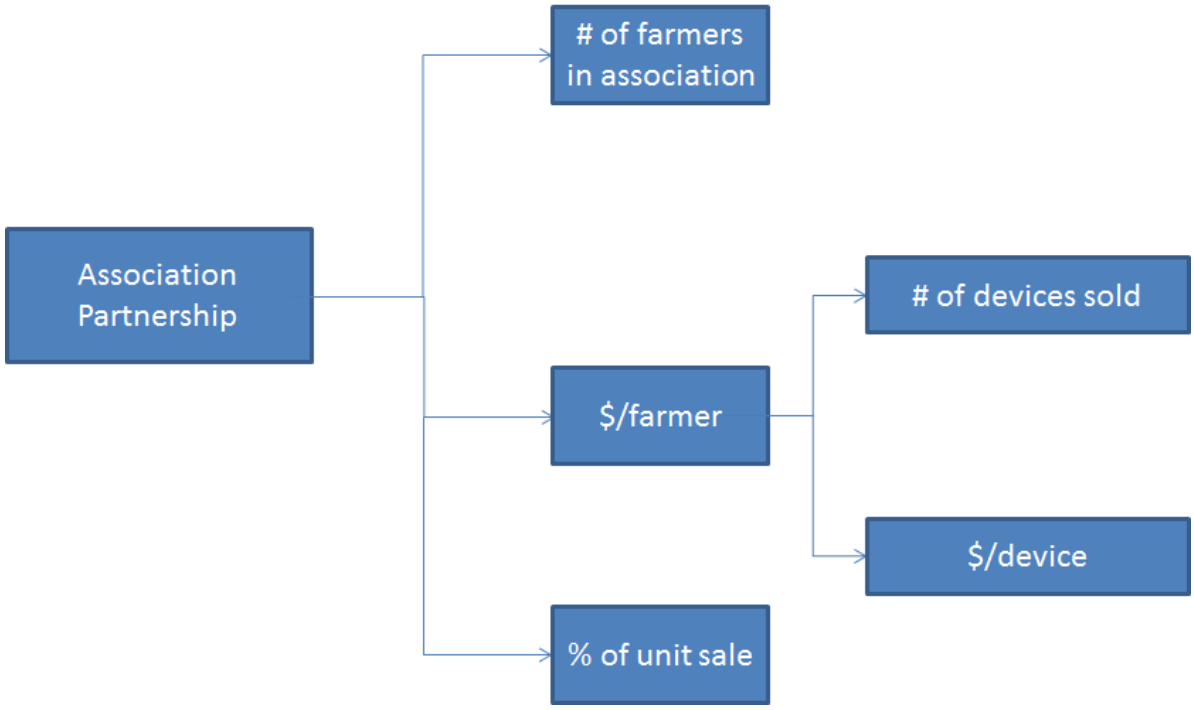
Babson College's Environmental and Sustainable Entrepreneurship (EPS) course investigates the places where a business touches the environment and society. Students research the lifecycle of a business, its products, and services, then brainstorm ways of making the process and interactions more sustainable for people, planet, and profit.

One potential scenario for SproutWave business operations was defined and analyzed in this class. The assumptions and assignments were designed in a way that provided unique insight to the business planning and big-scale sustainability aspects that are not the focus of ADE. Additional business models and options for deployment were developed, and the team explored how the different choices would impact the environment and the communities it touched. The following report addresses the most prominent conclusions for ADE from EPS.

Additional Business Models

1. Association Partnership Model

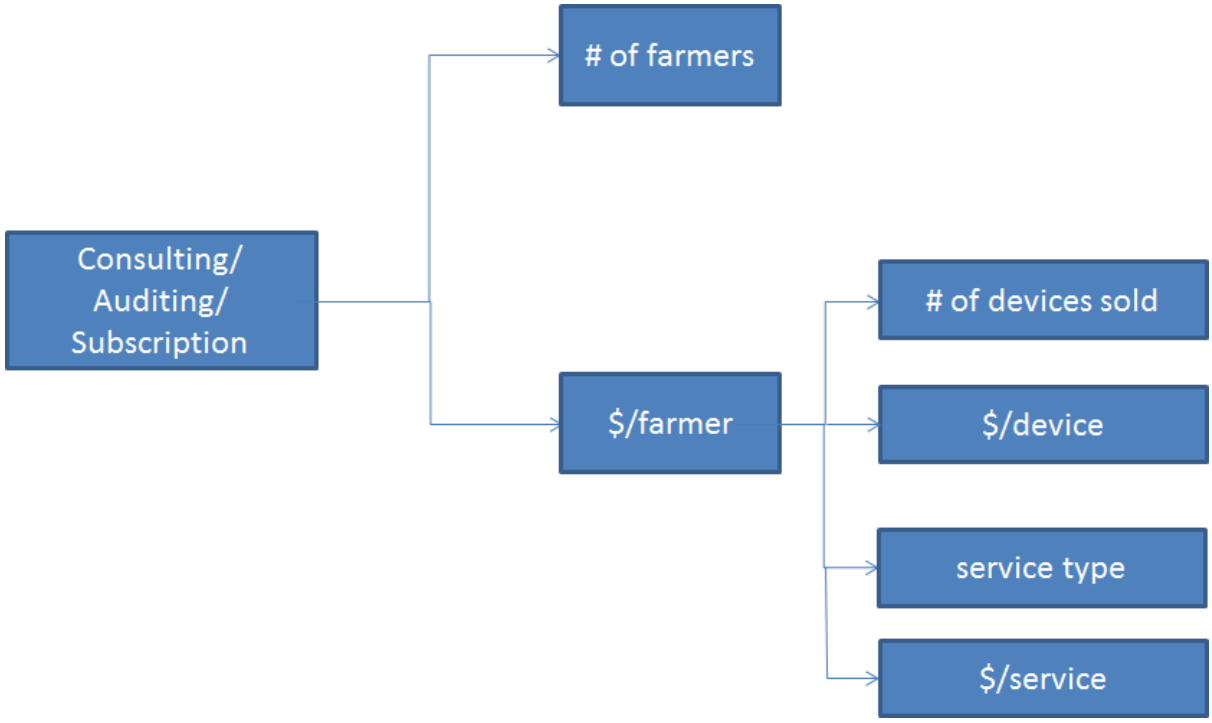
The Association Partnership Model consists of a model where many units are sold through partnerships with agricultural and farmer associations. The association will get a small cut for every product they sell to one of their association members. The revenue will be influenced by the number of farmers in the association that buy the product and the number of devices sold to each farmer, in addition to the price of each type of device. This time, the devices are ordered in bulk from an American manufacturer since in this model, associations are expected to create a list with a minimum of 25 farmers interested in the product. Since we are ordering in bulk, we may be able to encounter the same cost per unit as if we were outsourcing a single unit from China. This will reduce the impact on the environment. Although our margins per unit may be lower than in the Individual Unit Purchase Model, we believe we will have higher total margins. Agricultural and farmer associations have the power to communicate and influence very efficiently to farmers. Associations usually have a board of experts which makes them very credible. They also have the means (websites, media, conferences…) to communicate any new agricultural technology that could better a farm's performance. Farmers are less likely to be resistant to this approach because it is coming from their own association. This will also avoid any marketing costs and will accelerate sales. We believe this is the best revenue model.



2. Consulting/Auditing/Subscription Model

This model is designed for farmers who need help with the best installation of the product, a deeper interpretation of the data, and a continuous study of outer circumstances that may affect crop growth. This model can be incorporated in the Association Partnership Model and in the Individual Unit Purchase model. The revenue will be influenced by the number of farmers, the number of devices sold to each farmer, and the services provided to the farmer which include installation, consulting, and a subscription to a weekly journal. The price of the devices and the price of the services we deliver will also influence the revenue model. SproutWave's staff will use their expertise and know-how to install the product in the most efficient way so as to optimize the way the farm operates. They will also provide frequent studies of the farm's performance to the farmer which will let the farmer make better decisions to better manage his/her farm. The farmer can also be subscribed to a customized journal that informs him/her about outer circumstances that may affect growth and demand of their specific crops (weather conditions, market analysis…)

This model would be the least environmentally friendly, considering all the extra services provided but it will let the farmer manage their farm more efficiently. Farmers may be resistant to this model since they may not want to leave the performance of their farm in the hands of others. The nature of this model requires a lot of work and it supposes an added value for the farmer, therefore price would be higher and it could bring larger revenues per farmer than the other models.



Product Sustainability
A life cycle analysis (LCA) of high-tech products is extremely complex and can take many years, due to the large amount of parts, materials and processing techniques involved. In the meantime, products and processing technologies keep evolving, with the result that most life cycle analyses are simply outdated when they are published. Regular LCAs should be done, especially after changes to our hardware choices.

Each part of our system consists primarily of a motherboard and a case, and below we break out the numbers involved in processing and manufacturing a single unit.

The basic platform (or substrate) for a printed circuit board (PCB) is a fiberglass, copper, and epoxy-layered wafer-thin board. The copper is heavily processed so as to be layer-able, and the entire board is finished with a chemical etching process. The chemical slurry is usually recaptured and reused, because it is expensive to produce and the PCB manufacturers can save money that way.

The “things” that go on the PCB are called chips. Plastic, glass, silicon, and epoxy are layered and hardened over several days to make a sq. meter sheet that gets cut into over 10,000 chips. Super thin gold wire bonds chips to copper pins on top of the etched PCB, and then the whole thing is coated in epoxy.

In order to turn the circuit board and chips into a working computer, they need to be soldered together. Solder contains heavy metals (tin, antimony, silver) and whole process is energy intensive and requires high heat.

Case and housing for the computer is 4oz of steel and 2oz of ABS plastic. These amounts are slightly less for each sensor and outlet, around 2oz steel and 2oz plastic.

By far, the largest contributor to product sustainability is the circuit boards that allow interaction between all the parts of the system. More than just the raw materials, processing, and manufacturing, the end-of-life of the product (often trashed or recycled, containing a fully

functional board) is where that initial investment in PCB production becomes a big double negative for sustainability.

We should design modular circuit boards with open platforms so that we can recycle them as usable intelligence. At the very least allowing for the chip to be wiped clean of any proprietary functions and/or firmware would allow the PCB to be reused (in a SproutWave device or by an independent contractor). Additionally, local sourcing may be more sustainable as it requires less transportation, and we may be able to collect, wipe, and reuse PCBs already in existence for a double positive in both keeping it out of the waste stream, and eliminating the need for more raw materials for each new item.

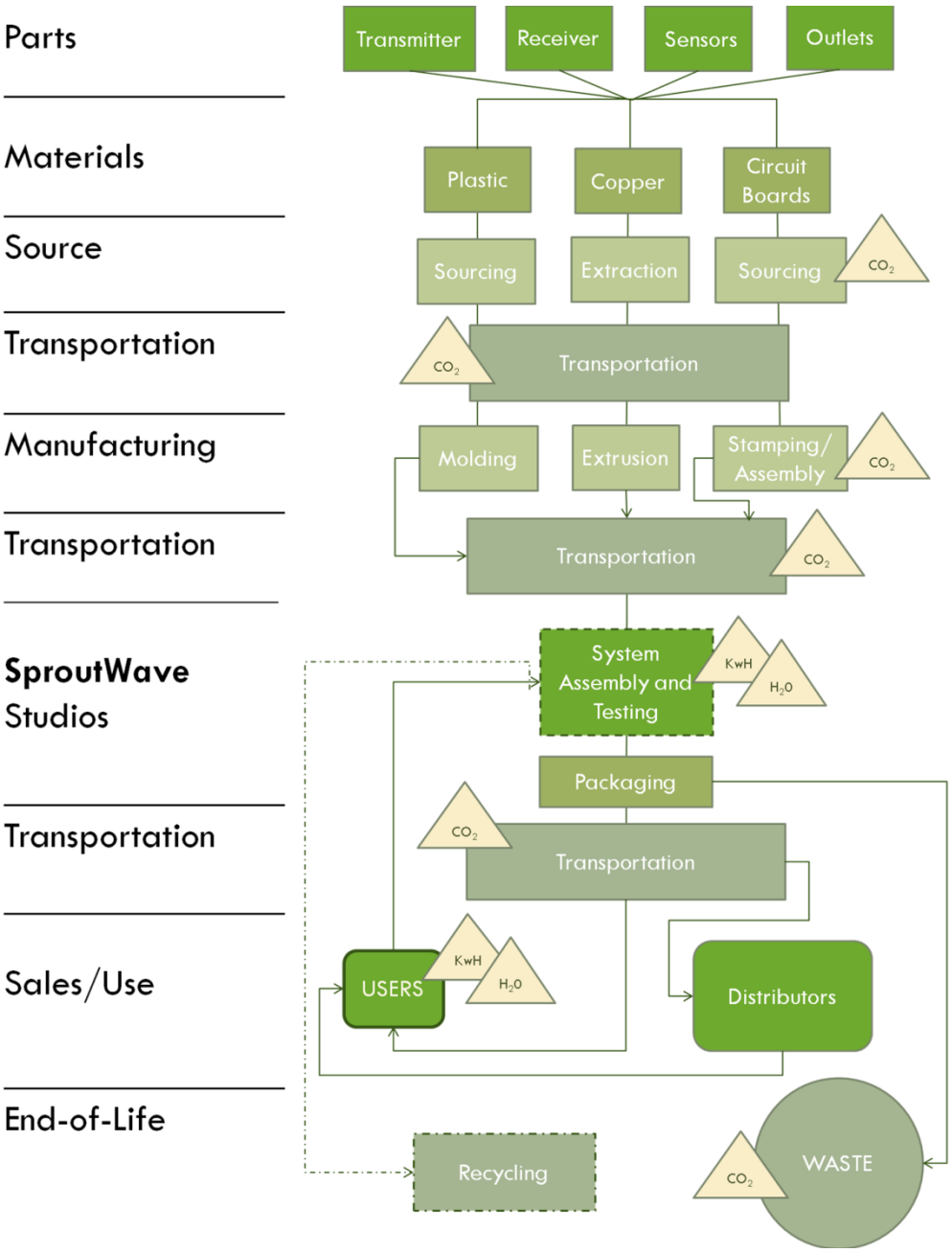
30 kg of CO2 are emitted from producing a single complex PCB, according to calculations from SimaPro[1]. With each base system including a computer, two sensors, and five outlets, we can estimate 8 PCBs needed per unit or 240kg. We assumed the plastic and steel for the cases would contribute an additional 20kg per piece or 160kg. This brings the total CO2 emissions from producing a single SproutWave system to 400kg or 0.4 metric tons

Business Sustainability

Since SproutWave, as any CCS, requires several electronic devices to operate (listed in the Carbon Footprint page), it does use electricity to function which has a minimal impact on the environment. Nonetheless, this is no waste of energy, but rather, and efficient use of energy. For instance, a farmer may have to move often from his/her house to the farm during non-working hours to turn on the cooling system in case of unexpected weather conditions during the summer. If the farm is an X minute drive away, driving the car for X minutes will most likely have a larger impact on the environment than using SproutWave for a whole year. Furthermore all devices our outsourced from China, this supposes another environmental impact. There are undoubtedly areas of improvement. The goal here though, would not be to reduce the energy consumption for SproutWave. This would not be too ambitious since the CCS itself does not consume too much energy compared to the other operations in the farm. The goal would be to build and design SproutWave so that it can intelligently optimize the energy use of the different systems in the farm such as fans and heaters.

$$\textit{Annual } CO_2 \textit{ emissions} = \frac{CO_2 \textit{ per gallon}}{MPG} \times \textit{miles} = \frac{8887}{28} \times 10000 = \textbf{\textit{3.2 metric tons}}$$

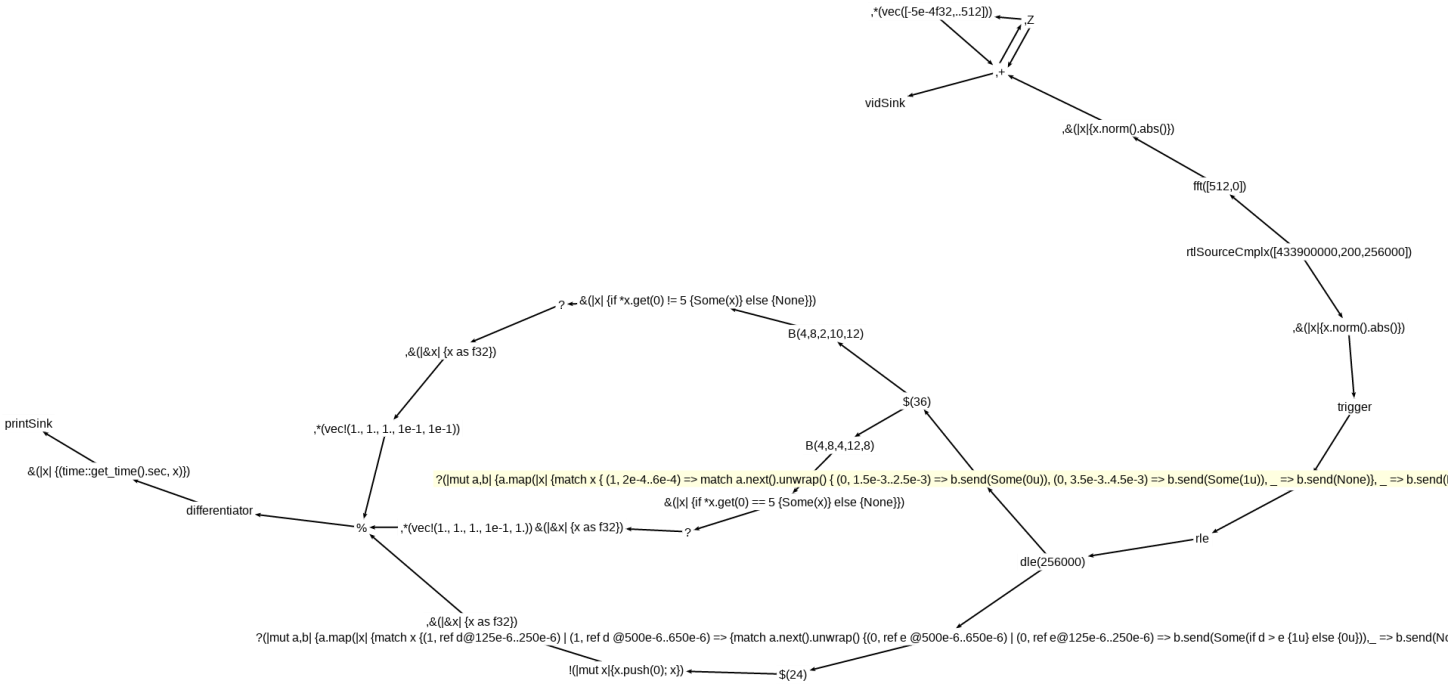
[1] http://www.energyreducingproducts.com/energysave/carbonfootprint.html



Conclusions

There is a great opportunity to target small farmers that have a limited budget and cannot afford a CCS of more than $100. Partnering with small farmer associations and organizations will speed up our market penetration and will let us fill in this infertile gap in the market. Although the manufacturing of our circuit boards is not the most environmentally sustainable, the CSS can be designed to also optimize the energy usage of the farm which will balance our environmental impact and to use fewer resources in production. We should use our R&D budget to explore alternatives or more sustainable methods of fabricating our current circuit boards. This product and business are feasible, and in all, we are confident about the profitability and sustainability of SproutWave.

10 Technical Appendicies


,*(vec([-5e-4f32,..512]))
_Z
,+
vidSink
,&(|x|{x.norm().abs()})
fft([512,0])
rtlSourceCmplx([433900000,200,256000])
,&(|x|{x.norm().abs()})
trigger
rle
dle(256000)
$(36)
B(4,8,2,10,12)
? &(|x| {if *x.get(0) != 5 {Some(x)} else {None}})
,&(|&x| {x as f32})
,*(vecl(1., 1., 1e-1, 1e-1))
printSink
&(|x| {(time::get_time().sec, x)})
differentiator
%
,*(vecl(1., 1., 1., 1e-1, 1.)) &(|&x| {x as f32})
,&(|&x| {x as f32})
?(|mut a,b| {a.map(|x| {match x {(1, ref d@125e-6..250e-6) | (1, ref d @500e-6..650e-6) => {match a.next().unwrap() {(0, ref e @500e-6..650e-6) | (0, ref e@125e-6..250e-6) => b.send(Some(if d > e {1u} else {0u})),_ => b.send(No
!(|mut x|{x.push(0); x})
$(24)
B(4,8,4,12,8)
?(|mut a,b| {a.map(|x| {match x { (1, 2e-4..6e-4) => match a.next().unwrap() { (0, 1.5e-3..2.5e-3) => b.send(Some(0u)), (0, 3.5e-3..4.5e-3) => b.send(Some(1u)), _ => b.send(None)}, _ => b.send(
&(|x| {if *x.get(0) == 5 {Some(x)} else {None}})
?

The JT65 Communications Protocol

Joe Taylor, K1JT

Abstract. JT65 is a digital protocol intended for Amateur Radio communication with extremely weak signals. It was designed to optimize Earth-Moon-Earth (EME) contacts on the VHF bands, and conforms efficiently to the established standards and procedures for such QSOs. JT65 includes error-correcting features that make it very robust, even with signals much too weak to be heard. This paper summarizes the technical specifications of JT65 and presents background information on its motivation and design philosophy. In addition, it presents some details of the implementation of JT65 within a computer program called WSJT, together with measurements of the resulting sensitivity and error rates.

1. Introduction

Spark gave way to continuous wave some eighty years ago. More or less by default, international Morse code with on-off keying has been the mode of choice for most amateur radio weak-signal work ever since. Morse is convenient, versatile, and readily encoded and decoded by humans. On-off keying is trivial to implement, and the required bandwidth is small. The choice has been an easy one.

It is easy to show, however, that neither the encoding nor the modulation of CW is optimum. When every dB of signal-to-noise ratio counts, as it does in amateur meteor-scatter and EME contacts, there are very good reasons to explore other options. Personal computers equipped with sound cards provide a golden opportunity for experimenting with the wide range of possibilities. The program WSJT[1,2,3] ("Weak Signal communications, by K1JT") is the result of my effort to introduce much more efficient coding and modulation schemes into amateur weak-signal communications. In the program's brief existence it has already become well known to nearly all weak-signal VHF/UHF operators, and is in regular use by many of them. On the VHF bands the overwhelming majority of all meteor-scatter QSOs and perhaps half of all EME QSOs are now being made with the help of WSJT.

The present paper describes JT65, one of the communications protocols supported by WSJT. JT65 is designed explicitly for communicating with extremely weak signals like those encountered on the EME path. Operational aspects of the program are described in the *WSJT User's Guide*[4]*;* here I will be concerned with a complete technical description of the protocol and a general description of the way it is implemented in WSJT.

Modern digital communication systems are based on the mathematics of information theory. This field essentially originated with two classic 1948 papers[5] in which Claude Shannon

---

[1] See the WSJT Home Page at http://pulsar.princeton.edu/~joe/K1JT.

[2] J. Taylor, K1JT, "WSJT: New Software for VHF Meteor-Scatter Communication," *QST* December 2001, pp. 36-41.

[3] J. Taylor, K1JT, "JT44: New Digital Mode for Weak Signals," *QST* June 2002, pp. 81-82

[4] The *WSJT 4.7 User's Guide* is available at http://pulsar.princeton.edu/~joe/K1JT/WSJT_User_470.pdf.

[5] Shannon, C. E., "A Mathematical Theory of Communicaton," *Bell System Tech. J.*,**27**, pp. 379–423 and 623–656, 1948.

proved that information can be conveyed over a noisy channel with arbitrarily low error rate and a throughput that depends only on channel bandwidth and signal-to-noise ratio (SNR). Achieving a low error rate at very low SNR requires the mathematical encoding of user information into a form that is compact yet includes carefully structured redundancy. Compactness is necessary in order to minimize transmitter power and maximize throughput; redundancy is needed to ensure message integrity on a noisy and variable channel.

To be transmitted by radio, an encoded message must be impressed onto a carrier wave using some form of modulation. The possibilities are almost limitless: information can be conveyed by varying the amplitude, frequency, or phase of a carrier, or any combination thereof. Commonly used digital modulation schemes include on-off keying (a limiting case of amplitude modulation), phase-shift keying, and frequency shift keying. The JT65 protocol uses 65-tone frequency shift keying with constant-amplitude waveforms and no phase discontinuities. This form of modulation is much more efficient than on-off keying, especially when combined with an optimal coding scheme. In addition, it is much more tolerant of frequency instabilities than phase-shift keying.

Section §2 of the paper begins with some background information that has helped to motivate the design philosophy of JT65, and Section §3 presents a high-level view of the overall system design. The protocol itself is defined §4–8 and in Appendix A, while Sections §9–12 describe the reception and decoding of a JT65 signal. The protocol specification completely defines the translation of any valid JT65 message into a waveform for transmission, and provides all information necessary for decoding a received JT65 signal. I include the essential details of how these tasks are actually carried out in WSJT. Different implementations of JT65, and especially the algorithms used for reception, are also feasible. I hope that this paper will motivate others to attempt this task, and that such efforts will lead to further improvements in the performance and operational convenience of this mode.

2. EME QSOs: Requirements and Procedures

Amateur Radio is a just-for-fun activity, and for many the fun has always included such goals as making contacts with all continents, all US states, and as many DXCC entities as possible. These goals are especially difficult on the EME path — and therefore, for many, all the more challenging and desirable. To make the game one that anybody can understand and play, it is necessary to agree on some basic ground rules.

When signals are reasonably strong and communication between skilled operators essentially error free, it is easy to judge whether a QSO has taken place. When a rare one shows up on the amateur HF bands, rapid-fire QSOs in the ensuing pile-up generally proceed something like the following exchange:

1.	CQ HC8N	
2.		K1JT
3.	K1JT 599	
4.		599 TU
5.	73 HC8N	

In this model contact K1JT never sends the callsign of the station he is working, because the situation has made this information implicit and moot. The signals may not be “S9” at either receiver, but no one really cares. After the exchange has taken place, both stations

confidently enter the QSO in their logs, and they may later exchange QSL cards to confirm that the contact took place.

In the VHF/UHF world, and especially when working over the EME path, signals are often very weak and communication between even the most skilled operators is far from error free. As a result, more rigorous standards need to be adopted for what constitutes a minimum legitimate QSO. Long-established rules hold that a valid contact requires each station to copy both complete callsigns, a signal report or some other piece of information, and explicit acknowledgment that all of this information has been received. These guidelines apply and work well for all types of weak-signal QSOs, whether by tropo, meteor scatter, EME, or other propagation modes, and with all types of equipment and signaling methods.

Following these guidelines closely, the minimal EME QSOs of savvy VHF operators generally proceed something like the following sequence:

1.	CQ SV1BTR ...	
2.		SV1BTR K1JT ...
3.	K1JT SV1BTR OOO ...	
4.		RO ...
5.	RRR ...	
6.		73 ...

For a scheduled QSO at prearranged time and frequency, transmission #1 is of course unnecessary. The ellipses (...) indicate repetition of messages, some form of which is nearly always used in EME contacts to help maximize chances of success. The "OOO" message component is a shorthand notation for a minimal signal report. It has an agreed-upon meaning that says, in effect, "your signals are readable at least some of the time, and I have copied both of our callsigns." Similarly, "RO" is a shorthand message conveying both signal report and acknowledgment. It means "I have copied both calls and my signal report, and your report is O". When K1JT receives the acknowledgment "RRR" sent by SV1BTR, the QSO is complete; but since SV1BTR does not yet know this, it is conventional to send "73" or some other end-of-contact information to signify "we are done."

Shorthand radio messages have been widely employed since the days of spark and land-line telegraphy; the familiar Q-signals are another universally understood type. They are simple forms of what in communication theory is called the "source encoding" of messages. The choice of "OOO…" (repeated sequences of three carrier-on intervals separated by short spaces, with a longer space after every third one) as the signal representing a positive signal report was made by wise and experienced CW operators who knew that with extremely weak signals, "dahs" are easier to copy than "dits".

3. System Design

Figure 1 presents the flow diagram of a modern digital communication system. For maximum efficiency at low signal-to-noise ratio, a user message is source encoded into a compact form having minimum redundancy. It is then augmented with mathematically defined redundancies which can enable full recovery of the message even if some parts are subsequently corrupted by noise or signal dropouts. This process is known as "forward error correction," or FEC. The encoded message, including its error-correcting information, is modulated onto a carrier. The resulting radio signal propagates over a channel that attenuates it, perhaps by 250 dB or more for an EME path, and adds noise as well as amplitude,

frequency, and phase-changing "path modulation." Upon reception the signal is demodulated and decoded, and the results presented to the user.

Except for the error-correcting enhancements, the flow diagram of Figure 1 describes traditional amateur CW communications just as well as modern digital techniques. In terms of the CW EME QSO outlined on the previous page, source encoding compresses the implied message "SV1BTR, this is K1JT, I have copied both of our calls" into the compact form "SV1BTR K1JT OOO". To provide some error-recovery capability and increase chances that the message will be copied, a CW operator repeats the compressed message many times during a timed transmission. To enhance the chances of copy even further, he may format the repetitions so as to transmit only calls for the first 75% of a transmission, followed by sending "OOO" repeatedly for the last 25%. He expects the receiving operator to know about these conventions, and to listen accordingly. All of these forms of source encoding help: the more that's known about the characteristics of a weak signal, the easier it is to copy. Under extremely marginal conditions, skilled operators listen for matches between what they hear and the types of message components they might reasonably expect. If a good match is found, message copy can be considered secure.

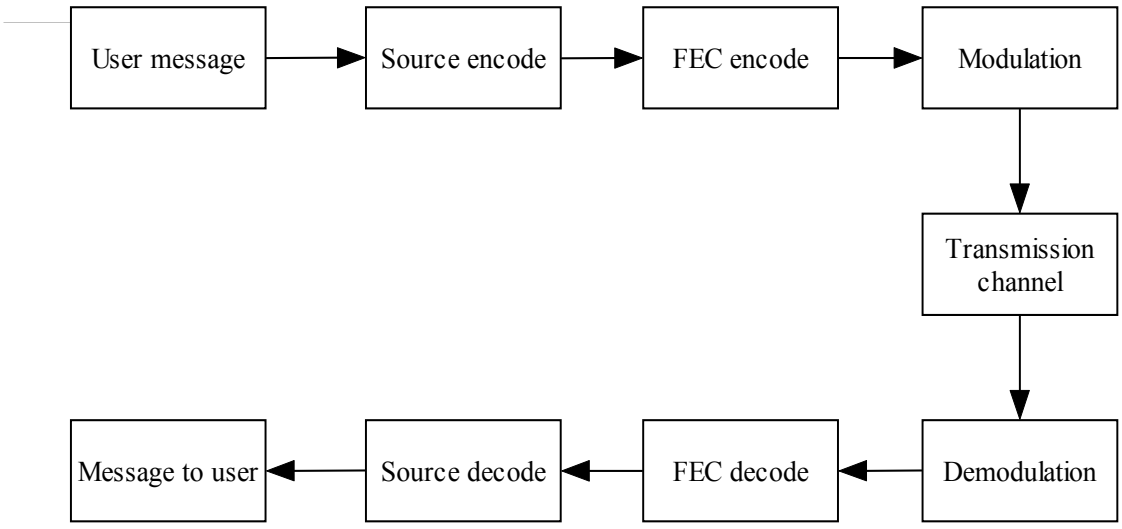


Fig. 1. – Schematic diagram of information flow in a digital communication system.

4. JT65 Source Encoding

JT65 uses exactly analogous techniques, starting out by making its transmitted messages compact and efficient. As described in the *WSJT 4.7 User's Guide*[4], the standard "Type 1" messages of JT65 consist of two callsigns, a grid locator, and an optional signal report — an enhanced form of messages 2 and 3 in the model QSO between SV1BTR and K1JT. The source encoder knows the rules by which standard amateur radio callsigns are constructed, and uses this information to minimize the required number of information bits. An amateur callsign consists of a one- or two-character prefix, at least one of which must be a letter, followed by a digit and a suffix of one to three letters. Within these rules, the number of possible callsigns is equal to 37×36×10×27×27×27, or somewhat over 262 million. (The

numbers 27 and 37 arise because in the first and last three positions a character may be absent, or a letter, or perhaps a digit.) Since 2^{28} is more than 268 million, 28 bits are enough to encode any standard callsign uniquely. Similarly, the number of 4-digit Maidenhead grid locators on earth is 180×180 = 32,400, which is less than 2^{15} = 32,768; so a grid locator requires 15 bits in a message. These important ideas for the efficient source encoding of EME messages were first suggested by Clark and Karn[6] in 1996.

Any Type 1 message can be source-encoded into 28+28+15= 71 bits, plus one more for the signal report. In comparison, sending the message "SV1BTR K1JT OOO" in Morse code requires 170 bits (where a bit is defined as the key-down dot interval), even without the grid locator. The JT65 message is much more compact than the CW message, while conveying significantly more information. In practice, the JT65 protocol encodes signal reports in another way and instead uses the 72nd bit to indicate that the message contains arbitrary text instead of callsigns and a grid locator. With a 43-character alphabet, the maximum plain-text message length is 13 (the largest integer less than 71 log 2/log 43). Subject to this limiting size, JT65 can transmit and receive *anything* in a message.

As indicated above, some 6 million of the possible 28-bit values are not needed for callsigns. A few of these slots have been assigned to special message components such as "CQ" and "QRZ". CQ may be followed by three digits to indicate a desired callback frequency. (If K1JT transmits on a standard calling frequency, say 144.120, and sends "CQ 113 K1JT FN20", it means that he will listen on 144.113 and respond there to any replies.) A numerical signal report of the form "–NN" or "R–NN" can be sent in place of a grid locator. The number NN must lie between 01 and 30. If required by licensing authorities, a country prefix or portable suffix may be attached[7] to one of the callsigns, as in ZA/PA2CHR or G4ABC/P. If this feature is used, the additional information is sent in place of the grid locator. Some remaining details of message encoding can be found in Appendix A, and a list of supported "add-on" prefixes and suffixes is presented in Appendix B.

5. Forward Error Correction

After being compressed into 72 bits, a JT65 message is augmented with 306 uniquely defined error-correcting bits. The FEC coding rate is thus $r = 72/378 = 0.19$; equivalently one might say that each message is transmitted with a "redundancy ratio" of 378/72 = 5.25. With a good error-correcting code, however, the resulting performance and sensitivity are far superior to those obtainable with simple five-times message repetition. The high level of redundancy means that JT65 copes extremely well with QSB. Signals that are discernible to the software for as little as 10 to 15 s in a transmission can still yield perfect copy.

The source of this seemingly mysterious "coding gain" is not difficult to understand. With 72 bits the total number of possible user messages is 2^{72}, slightly more than 4.7×10^{21}. The number of possible patterns of 378 bits is a vastly larger number, 2^{378}, in excess of 6×10^{113}. With a one-to-one correspondence between 72-bit user messages and 378-bit "codewords,"

---

[6] Clark, T. W3IWI, and Karn, P., KA9Q, "EME 2000: Applying Modern Communications Technologies to Weak Signal Amateur Operations," *Proc. Central States VHF Society,* 1996.

[7] Callsign prefixes and suffixes were accommodated in a somewhat different way in WSJT versions 4.9.2 and earlier.

or unique sequences of 378 bits, it is clear that only a tiny fraction of the available sequences need to be used in the code. The sequences chosen are those that are "as different from one another as possible," in a mathematically rigorous sense.

A huge variety of efficient error correcting codes are known and understood mathematically. Among the best known are the Reed Solomon codes, used to produce the extremely low error rates characteristic of modern CD-ROMs and hard disk drives. For JT65 I chose the Reed Solomon code RS(63,12), which encodes each 72-bit user message into 63 six-bit "channel symbols" for transmission. Every codeword in this code differs from every other one in at least 52 places — which, in a nutshell, is why the code is so powerful. Even at very low SNR, distinct sequences are very unlikely to be confused with one another.

```
Message #1:  G3LTF DL9KR JO40
Packed message, 6-bit symbols:  61 37 30 28  9 27 61 58 26  3 49 16
Channel symbols, including FEC:
   14 16  9 18  4 60 41 18 22 63 43  5 30 13 15  9 25 35 50 21  0
   36 17 42 33 35 39 22 25 39 46  3 47 39 55 23 61 25 58 47 16 38
   39 17  2 36  4 56  5 16 15 55 18 41  7 26 51 17 18 49 10 13 24

Message #2:  G3LTE DL9KR JO40
Packed message, 6-bit symbols:  61 37 30 28  5 27 61 58 26  3 49 16
Channel symbols, including FEC:
   20 34 19  5 36  6 30 15 22 20  3 62 57 59 19 56 17 35  2  9 41
   10 23 24 41 35 39 60 48 33 34 49 54 53 55 23 24 59  7  9 39 51
   23 17  2 12 49  6 46  7 61 49 18 41 50 16 40  8 45 55 45  7 24

Message #3:  G3LTF DL9KR JO41
Packed message, 6-bit symbols:  61 37 30 28  9 27 61 58 26  3 49 17
Channel symbols, including FEC:
   47 27 46 50 58 26 38 24 22  3 14 54 10 58 36 23 63 35 41 56 53
   62 11 49 14 35 39 60 40 44 15 45  7 44 55 23 12 49 39 11 18 36
   26 17  2  8 60 44 37  5 48 44 18 41 32 63  4 49 55 57 37 13 25
```

Fig. 2. – Three JT65 messages shown as they appear to the user; in 72-bit packed form, displayed as 12 × 6-bit symbol values; and as FEC-enhanced sequences of 63 × 6-bit channel symbols. The channel symbols are ready to be transmitted by means of 64-tone FSK, with each symbol value corresponding to a distinct tone.

As an example, the encoded sequences for three nearly identical messages are illustrated in Figure 2. Lines labeled "packed message" show each source-encoded, 72-bit user message as a sequence of twelve 6-bit symbols. Reading from left to right, one can see that the fifth numerical symbol changes from 9 to 5 when the last letter in the first callsign changes from F to E. The final packed symbol changes from 16 to 17 when the grid locator changes from JO40 to JO41. Otherwise, the three packed messages are identical. On the other hand, the three fully encoded sequences of channel symbols appear to be almost entirely different from one another — so different that *there is virtually no chance whatsoever that, if it is decodable at all, a noise-corrupted version of one of these messages would ever be misconstrued as one of the others*. The full and exact user message has a high probability of being received, even if the key-down SNR is as low as 2 to 6 dB in 2.7 Hz bandwidth (or –28 to –24 dB in 2500

Hz, the conventional reference bandwidth used in WSJT). This statement can be quantified by explicit measurements of transmission error rates as a function of SNR, and such measurements are summarized for JT65 in Appendix C.

6. Interleaving and Gray Coding

After encoding, the order of JT65 symbols is permuted by writing them row-by-row into a 7×9 matrix, and reading them out column-by-column. I was studying FEC for the first time when JT65 was being designed, and I mistakenly believed that scrambling the symbol order would give the system greater immunity to signal dropouts. In fact, it does not; but since its effect is quite harmless, the procedure has been left intact to preserve the integrity of JT65 signals over subsequent program versions. The re-ordered symbols are converted from binary to Gray-code representation, which makes JT65 somewhat more tolerant of frequency instabilities.

7. Shorthand Messages

Like the CW methods described earlier, JT65 uses special signal formats to convey frequently used messages in a robust and efficient way. Three such messages are presently defined. They correspond exactly to the transmissions numbered 4, 5, and 6 in the model CW QSO between SV1BTR and K1JT, conveying the messages "RO", "RRR", and "73". Instead of keying a single-frequency carrier on and off according to a pattern like di-dah-dit, dah-dah-dah, …, JT65 sends "RO" by transmitting two alternating tones with specified frequencies and a specified keying rate. Such waveforms are easy to recognize and to distinguish from one another, as well as from "normal" JT65 messages. Indeed, as many users have discovered, the shorthand messages of JT65 are readily decodable by human operators using sight or sound, as well as by computer.

8. Synchronization and Modulation

JT65 uses one-minute T/R sequences and requires tight synchronization of time and frequency between transmitter and receiver. Typical amateur equipment cannot accomplish this task with sufficient accuracy in open-loop fashion, so a JT65 signal must carry its own synchronizing information. A pseudo-random "sync vector" is therefore interspersed with the encoded information bits. It allows accurate calibration of relative time and frequency errors, thereby establishing a rigorous framework within which the decoders can work. In addition, it enables the averaging of successive transmissions so that decoding is possible even when signals are too weak to accomplish it in a single transmission. The synchronizing signal is so important that (except in shorthand messages) half of every transmission is devoted to it.

A JT65 transmission is divided into 126 contiguous time intervals, each of length 0.372 s (4096 samples at 11025 samples per second). Within each interval the waveform is a constant-amplitude sinusoid at one of 65 pre-defined frequencies, and frequency changes between intervals are accomplished in a phase-continuous manner. A transmission nominally begins at t = 1 s after the start of a UTC minute and finishes at t = 47.8 s. The synchronizing tone is at frequency 1270.5 Hz and is normally sent in each interval having a "1" in the pseudo-random sequence reproduced at the top of Figure 3. The sequence has the desirable mathematical property that its normalized autocorrelation function falls from 1 to nearly 0 for all non-zero lags. As a consequence, it makes an excellent synchronizing vector.

Encoded user information is transmitted during the 63 intervals not used for the sync tone. Each channel symbol generates a tone at frequency 1270.5 + 2.6917 (N+2) m Hz, where N is the integral symbol value, $0 \le N \le 63$, and m assumes the values 1, 2, and 4 for JT65 sub-modes A, B, and C. The signal report "OOO" is conveyed by reversing sync and data positions in the pseudo-random sequence. Because normal messages depend on tight synchronization, they can be initiated only at the beginning of a UTC minute.

```
1,0,0,1,1,0,0,0,1,1,1,1,1,1,0,1,0,1,0,0,0,1,0,1,1,0,0,1,0,0,
0,1,1,1,0,0,1,1,1,1,0,1,1,0,1,1,1,1,0,0,0,1,1,0,1,0,1,0,1,1,
0,0,1,1,0,1,0,1,0,1,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,1,
0,1,0,0,1,0,1,1,0,1,0,1,0,1,0,0,1,1,0,0,1,0,0,1,0,0,0,0,1,1,
1,1,1,1,1,1
```

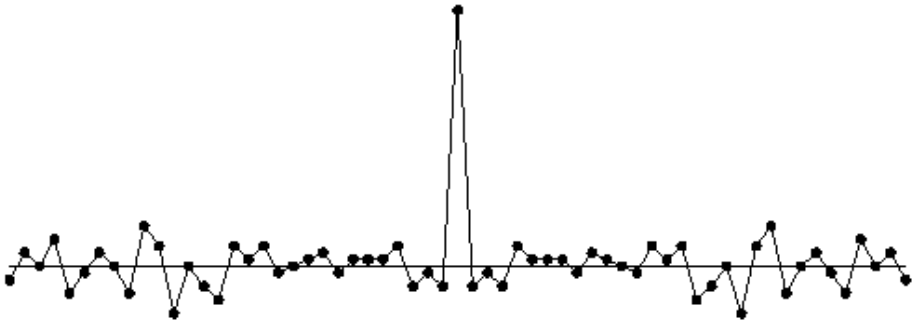


Fig. 3. – The pseudo-random sequence used in JT65 as a "synchronizing vector," and a graphical representation of its autocorrelation function. The isolated central correlation spike serves to synchronize time and frequency between transmitting and receiving stations.

Shorthand messages dispense with the sync vector and use intervals of 1.486 s (16,384 samples) for the alternating tones. The lower frequency is always 1270.5 Hz, the same as that of the sync tone. The frequency separation is 26.917 nm Hz with n = 2, 3, 4 for the messages RO, RRR, and 73. By the time shorthand messages become relevant in a QSO, the frequency offset between transmitter and receiver has already been measured with high accuracy. As a consequence, these messages can be securely identified by the operator as coming from the station whose callsign was recently decoded. Accurate time synchronization is not required for shorthand messages, so they may be started at any time during a transmission.

By now it should be clear that JT65 does not transmit messages character by character, as done in Morse code. Instead, whole messages are translated into unique strings of 72 bits, and from those into sequences of 63 six-bit symbols. These symbols are transmitted over a radio channel; some of them may arrive intact, while others are corrupted by noise. If enough of the symbols are correct (in a probabilistically defined sense), the full 72-bit compressed message can be recovered *exactly*. The decoded bits are then translated back into the human-readable message that was sent. The coding scheme and robust FEC assure that messages are never received in fragments. Message components cannot be mistaken for one another, and callsigns are never displayed with a few characters missing or incorrect. There is no chance for the letter O or R in a callsign to be confused with a signal report or an acknowledgment, or for a fragment of a callsign like N8CQ or a grid locator like EM73 to be

misinterpreted. If your sked partner does not show and another station calls in his place, you will never conclude mistakenly than the schedule was kept as intended.

9. Reception and Demodulation

Within WSJT, a received JT65 signal is converted to baseband and analyzed using a sequence of well known DSP techniques. The process begins with an audio signal in the approximate frequency range 0–3 kHz, digitized at the nominal rate 11025 samples per second. The digital signal is low-pass filtered and downsampled by a factor of two. Power spectra are computed from discrete Fourier transforms of sliding 2048-sample blocks and examined for presence of the pseudo-random sync pattern. Detection and "peaking up" on the sync pattern establishes the required frequency and time offsets, which may include Doppler shift and EME path delays as well as errors in frequency calibration and clock settings. The synchronizing accuracy is typically around 1.5 Hz in frequency and 0.03 s in time. Once "sync" has been established, the program re-measures the sync-tone frequency over small groups of tone intervals and fits a smooth curve to the results, thereby enabling the tracking and compensation of small frequency drifts. Coherent phase tracking between symbols is not required.

With accurate sync information in hand, the program computes a 64-bin spectrum for each of the 63 channel symbols. These spectra have resolution 2.7m Hz (e.g., 5.4 Hz for sub-mode JT65B, m = 2), and with very weak signals they are essentially noise-like in form. Many of the individual data tones may not be detectable above the noise. On average, however, in each tone interval the one frequency bin containing signal will have greater amplitude than the others. Using the known statistical properties of random Gaussian noise, WSJT computes the probability that a symbol was transmitted with each one of the possible values. This probabilistic information, based on measured spectra of the synchronized symbols, is the basic received information. After Gray coding and symbol interleaving have been removed, the probabilities are passed on to the decoder.

10. Reed Solomon Decoder

Even a small error-correcting code like RS(63,12) can be very difficult to "invert" or decode efficiently. The basic problem is this: given the measured spectra for each of the 63 channel symbols, is there a unique 72-bit sequence that can be confidently identified as the user's message? In principle, one might encode each of the 2^{72} possible user messages and correlate the results against the received spectra, looking for a match. Such an approach is quite impractical, however: a simple estimate reveals that with today's 3 GHz computer, unlimited memory, and a very efficient program, it would take about 200 million years to decode a single received message this way.

Reed Solomon codes are economically important because well defined mathematical algorithms exist for decoding them. The algorithms vary in complexity and in how closely they approach the ideal sensitivity of the method just described. Since program version 4.5, WSJT has used an algorithm that represents the state of the art in Reed Solomon decoding. It

is based on a research paper by Ralf Koetter and Alexander Vardy[8], and uses computer code licensed from their company, CodeVector Technologies. Furnished with soft-decision probabilistic information on received symbol values, this decoder produces a clear result for every transmission analyzed. With very high confidence, it returns either the 72 bits of the transmitted message or else a flag indicating "no result".

Error rates for the WSJT decoders have been carefully measured as a function of signal level. The results are summarized in Appendix C. Briefly stated, the K-V decoder exhibits a steep transition from "nearly always decoding" to "nearly always failing" as the signal-to-noise ratio decreases from about –23 to –25 dB (for JT65B) on the WSJT scale. The results further show that with "clean" data (additive Gaussian noise, and perhaps fading, but no interference from other signals), false decodes from the K-V decoding algorithm on RS(63,12) are so rare that you will hardly ever see one.

11. Deep-Search Decoder

What if the K-V decoder fails to produce a result? Can anything further be done? Life is too short to consider correlating all 2^{72} possible user messages in search of a match, but the number of unique messages transmitted in real EME QSOs is actually very much smaller than 2^{72}, and the ones you are most interested in are fewer still. If the more plausible and more interesting messages are tested first — more or less in the same way that one does when copying very weak CW — and if the search algorithm is instructed to "time out" if no match is found after a reasonable time, the brute-force computational approach described above can be made practical. In WSJT, a procedure I call the "deep search" algorithm attempts to do just this.

The deep search starts with a list of plausible callsigns and grid locators. Such lists have long been maintained, both mentally and in hard copy, by most EME operators. They can be of great help when trying to determine which station might be transmitting a weak CQ, answering your own CQ, or tail-ending your last QSO. In the WSJT deep search decoder, each list entry is paired with "CQ" and with the home callsign of the WSJT user, thereby creating hypothetical test messages. If N_c calls are present in the list, approximately $2N_c$ messages will be generated, fully encoded, and the channel symbols tested for good match with the observed spectra. You can define the list of likely callsigns in any way you choose. An example file is provided with WSJT, containing the calls of nearly 5000 worldwide stations known to have been active in weak-signal work on the VHF/UHF bands. Knowledgeable JT65 users maintain their own files, adding or deleting calls as they deem appropriate.

In effect, your callsign database defines a set of matched filters, custom designed for your station and tuned for optimum sensitivity to a subset of the messages you might reasonably expect to receive. The deep search is not sensitive to messages with callsigns not in the database, or arbitrary plain text, or anything besides "CQ" or your own call in the first message field. Such messages will be decoded with the already remarkable sensitivity of the K-V algorithm. However, for any message within the defined subset, the deep search decoder provides about 4 dB more sensitivity while still maintaining a low error rate. It

[8] Koetter, R., and Vardy, A., "Soft-Decision Algebraic Decoding of Reed Solomon Codes," in *Proceedings of the IEEE International Symposium on Information Theory*, p. 61, 2000.

should be obvious that those 4 dB are essentially equivalent to the widely recognized "schedule gain" that CW operators can experience when copying familiar calls or making pre-arranged contacts.

12. Decoding Shorthand Messages

In addition to seeking a synchronizing tone modulated with the expected pseudo-random pattern, WSJT searches for alternating tones having the specified modulation of a JT65 shorthand message. Frequencies are measured and compared with that of the sync tone in a previous transmission, and a test is made to be sure that the modulation follows the specified square-wave cycle. If the frequencies and modulation match, and if the amplitude exceeds a preset threshold, a shorthand message detection is declared. Because of the close frequency and timing tolerances, a low detection threshold can be set while still maintaining a very low rate of false positives. Measured sensitivity curves for shorthand messages are presented in Appendix C, along with those for the K-V algorithm and the deep search decoder.

13. Operator Responsibilities and Message Integrity

QSOs made with any of the WSJT modes, including JT65, require active user participation at all stages. In the presence of birdies, QRM, QRN, or other anomalies such as multipath signal distortions, operator involvement is necessary to avoid mistakes in interpreting program output. Most operators find that they acquire the necessary skills easily, while making their first few JT65 contacts.

In connection with the guidelines for valid QSOs outlined in Section 2, it is worth making special mention of a particular feature of JT65. Contacts made with WSJT are inherently self-documenting. When a JT65 QSO is successfully completed, both operators *know* that the requisite information has been exchanged. Moreover, if desired, they have the recorded wave files to prove it. These files provide a "bit trail," an essentially incorruptible proof of copy that anyone could examine. After especially interesting or difficult QSOs, recorded waveforms and screen images are often exchanged by email. I have accumulated a large library of JT65 wave files from my own QSOs, and by monitoring the bands, as well as many sent to me by others. These files have proven extremely valuable for refining WSJT's algorithms for optimum sensitivity and minimum error rate, under real-world conditions. Further progress will surely be made in these areas, in years to come.

14. On-the-Air Experience

The first usable version of JT65 was finished in November 2003. Early on-the-air tests with N3FZ quickly confirmed my expectation that JT65 would become a major new weapon in the arsenal of VHF/UHF weak-signal enthusiasts. The practical advantages of error-correcting codes for weak-signal amateur radio communication were very plainly evident. Little wonder, I realized, that NASA always transmits its deep-space photographs back to Earth using tight source encoding and strong FEC. In deep-space communications, every dB of improved sensitivity can save millions of dollars that would otherwise have to be spent on larger antennas or more transmitter power.

Definition of the JT65 protocol has evolved only in minor ways since the first test transmissions. Meanwhile, the decoders have been steadily improved, producing sizable advances in on-the-air performance. I have no way of knowing how many EME QSOs have

been made with JT65, but the number is surely in the many thousands. Users have not hesitated to report program bugs or suggest operational improvements, and WSJT has greatly benefited from such feedback. A sizable new group of EME enthusiasts has sprung up, attracted by the fact that JT65 QSOs can be accomplished with much more modest setups than required for traditional methods. Hundreds of JT65 EME QSOs have been made by stations running 150 W to a single yagi on the 2 m band, and QSOs with "big gun" stations have been made with as little as 5 W. Even 50 MHz EME QSOs, long considered among the most difficult of feats, have become a common occurrence.

15. Looking Ahead

I do not foresee the need for major revision or expansion of the JT65 technical specification. However, I can think of many ways in which the implementation of JT65 might be improved. To start with, received audio data should be processed as it comes in, rather than in "batch mode" after the whole reception period is complete. This would permit having a native real-time spectral display, and I can imagine an option to allow "early decoding" of signals after 20 or 30 s of received data have been acquired. I have learned that some sound cards exhibit errors as large as 0.6% in their sampling rates. The JT65 decoders presently in WSJT do not attempt to correct for such errors, and sensitivity suffers unnecessarily. A better job of detecting and suppressing interference can certainly be done. The algorithm presently used to track frequency drifts of the desired signal can be improved. Explicit tracking of Doppler-induced frequency changes is certainly desirable, especially at 432 and 1296 MHz. More accurate control of the timing of transmit/receive sequences would help, and might be possible even under Windows. Execution speed of the decoding procedures can be improved… and the list goes on and on. Perhaps others will take up the challenge to undertake some of these improvements, or will think of other enhancements that will be even more significant.

Appendix A: Details of Message Encoding

As described in Sections §4–6, JT65 message encoding takes place in several stages. A user's message is first "source encoded" into a compact form requiring just 72 bits. The bits are packed into twelve 6-bit information symbols, and a Reed Solomon encoder adds 51 parity symbols. The 63 channel symbols are interleaved, Gray coded, and transmitted using 64-tone frequency shift keying. A synchronizing vector is sent at a 65th frequency, two tone intervals below the lowest data tone.

Some arbitrary choices define further details of message packing and the ordering of channel symbols. To make it easy for others to implement the JT65 protocol, these things are best described with actual source code examples. Appended below is a Fortran program that can easily be compiled under Linux. Only the main program is listed here; the full source code, including necessary subroutines and a Linux makefile, can be downloaded from pulsar.princeton.edu/~joe/K1JT/JT65code.tgz. The compiled program accepts a JT65 message (enclosed in quotes on the command line) and responds with the packed message and channel symbols as six-bit values. Examples of program output were presented in Figure 3 and described in Section §5.

```
      program JT65code


C  Provides examples of message packing, bit and symbol ordering,
C  Reed Solomon encoding, and other necessary details of the JT65
C  protocol.

      character*22 msg0,msg,decoded,cok*3
      integer dgen(12),sent(63)

      nargs=iargc()
      if(nargs.ne.1) then
         print*,'Usage: JT65code "message"'
         go to 999
      endif

      call getarg(1,msg0)                !Get message from command line
      msg=msg0

      call chkmsg(msg,cok,nspecial,flip)  !See if it includes "OOO" report
      if(nspecial.gt.0) then              !or is a shorthand message
         write(*,1010)
 1010    format('Shorthand message.')
         go to 999
      endif

      call packmsg(msg,dgen)              !Pack message into 72 bits
      write(*,1020) msg0
 1020 format('Message:   ',a22)           !Echo input message
      if(and(dgen(10),8).ne.0) write(*,1030) !Is the plain text bit set?
 1030 format('Plain text.')
      write(*,1040) dgen
 1040 format('Packed message, 6-bit symbols: ',12i3) !Print packed symbols

      call packmsg(msg,dgen)              !Pack user message
      call rs_init                        !Initialize RS encoder
      call rs_encode(dgen,sent)           !RS encode
      call interleave63(sent,1)           !Interleave channel symbols
      call graycode(sent,63,1)            !Apply Gray code

      write(*,1050) sent
 1050 format('Channel symbols, including FEC:'/(i5,20i3))
      call unpackmsg(dgen,decoded)        !Unpack the user message
      write(*,1060) decoded,cok
 1060 format('Decoded message: ',a22,2x,a3)

 999  end
```

Appendix B: Supported Callsign Prefixes and Suffixes

Callsign prefixes and suffixes supported by JT65 are listed in the file `pfx.f` included in the source code archive at pulsar.princeton.edu/~joe/K1JT/JT65code.tgz, as described in Appendix A. Supported suffixes include /P and /0 through /9, while the full prefix list is appended below. Additional prefixes and suffixes could be added to the list in the future. Space for 450 prefixes has been reserved by not supporting any grid locators within 5° of the North Pole.

```
1A    1S    3A    3B6   3B8   3B9   3C    3C0   3D2   3D2C  3D2R  3DA   3V    3W    3X
3Y    3YB   3YP   4J    4L    4S    4U1I  4U1U  4W    4X    5A    5B    5H    5N    5R
5T    5U    5V    5W    5X    5Z    6W    6Y    7O    7P    7Q    7X    8P    8Q    8R
9A    9G    9H    9J    9K    9L    9M2   9M6   9N    9Q    9U    9V    9X    9Y    A2
A3    A4    A5    A6    A7    A9    AP    BS7   BV    BV9   BY    C2    C3    C5    C6
C9    CE    CE0X  CE0Y  CE0Z  CE9   CM    CN    CP    CT    CT3   CU    CX    CY0   CY9
D2    D4    D6    DL    DU    E3    E4    EA    EA6   EA8   EA9   EI    EK    EL    EP
ER    ES    ET    EU    EX    EY    EZ    F     FG    FH    FJ    FK    FKC   FM    FO
FOA   FOC   FOM   FP    FR    FRG   FRJ   FRT   FT5W  FT5X  FT5Z  FW    FY    M     MD
MI    MJ    MM    MU    MW    H4    H40   HA    HB    HB0   HC    HC8   HH    HI    HK
HK0A  HK0M  HL    HM    HP    HR    HS    HV    HZ    I     IS    IS0   J2    J3    J5
J6    J7    J8    JA    JDM   JDO   JT    JW    JX    JY    K     KG4   KH0   KH1   KH2
KH3   KH4   KH5   KH5K  KH6   KH7   KH8   KH9   KL    KP1   KP2   KP4   KP5   LA    LU
LX    LY    LZ    OA    OD    OE    OH    OH0   OJ0   OK    OM    ON    OX    OY    OZ
P2    P4    PA    PJ2   PJ7   PY    PY0F  PT0S  PY0T  PZ    R1F   R1M   S0    S2    S5
S7    S9    SM    SP    ST    SU    SV    SVA   SV5   SV9   T2    T30   T31   T32   T33
T5    T7    T8    T9    TA    TF    TG    TI    TI9   TJ    TK    TL    TN    TR    TT
TU    TY    TZ    UA    UA2   UA9   UK    UN    UR    V2    V3    V4    V5    V6    V7
V8    VE    VK    VK0H  VK0M  VK9C  VK9L  VK9M  VK9N  VK9W  VK9X  VP2E  VP2M  VP2V  VP5
VP6   VP6D  VP8   VP8G  VP8H  VP8O  VP8S  VP9   VQ9   VR    VU    VU4   VU7   XE    XF4
XT    XU    XW    XX9   XZ    YA    YB    YI    YJ    YK    YL    YN    YO    YS    YU
YV    YV0   Z2    Z3    ZA    ZB    ZC4   ZD7   ZD8   ZD9   ZF    ZK1N  ZK1S  ZK2   ZK3
ZL    ZL7   ZL8   ZL9   ZP    ZS    ZS8
```

Appendix C: Measured Sensitivity and Error Rates

The JT65 protocol can be defined once and for all, but on-the-air performance depends on a particular software implementation of the decoder. As outlined in §9–12, version 4.9 of WSJT does its JT65 decoding in three phases: a soft-decision Reed Solomon decoder, the deep search decoder, and the decoder for shorthand messages. Section §13 emphasizes that in circumstances involving birdies, atmospherics, or other interference, operator interaction is an essential part of the decoding process. The operator can enable a "Zap" function to excise birdies, a "Clip" function to suppress broadband noise spikes, and a "Freeze" feature to limit the frequency range searched for a sync tone. Having used these aids and the program's graphical and numerical displays appropriately, the operator is well equipped to recognize and discard any spurious output from the decoder.

Under normal conditions in which the transmission channel can be characterized by simple attenuation, the addition of white Gaussian noise, and perhaps multiplication by a "Rayleigh fading" coefficient, the sensitivities and error rates of the decoders can be accurately measured. A software simulator for doing this was written for the Linux platform as the first (and very essential) part of WSJT program development. The simulator can generate digitized waveforms for any WSJT mode and inject them into band-limited Gaussian noise with a specified signal-to-noise ratio and optional fading characteristics. The resulting audio

files can be saved in WAV format, then opened and decoded in WSJT. They can also be decoded directly within the simulator, using code identical to the WSJT decoder but compiled for Linux.

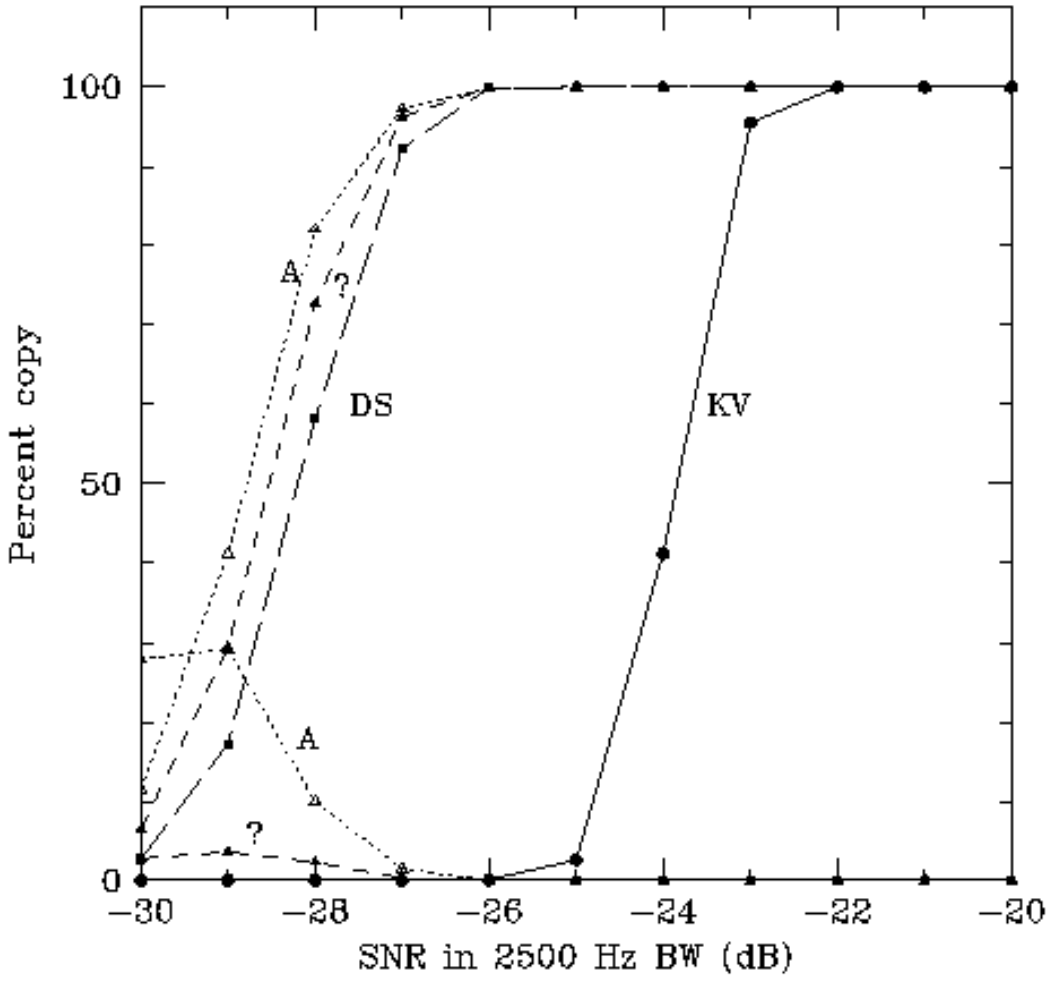


Fig. 4.– Measured rates of copy as a function of SNR for JT65B. The curve labelled KV refers to the Koetter-Vardy algorithm; DS refers to the deep search algorithm. The rate of false decodes for the KV algorithm is too small to measure; for the DS algorithm the rate of "hard errors" was about 0.03%, too small to show on this graph. Curves labelled "?" and "A" at the lower left give the deep-search soft-error rates for decoded messages marked "?" and when "Aggressive decoding" has been requested.

Several hundred thousand simulated JT65 transmissions have been tested in this way — first as a means of debugging and fine-tuning the decoders, and later as a way to measure the sensitivity and error rates of the finished program. Results of the simulations are summarized in Figures 4 and 5. To create Figure 4, 1000 simulated transmissions were generated and tested for each of the levels SNR = –30, –29, … –20 dB, using standard JT65 messages consisting of two callsigns and a grid locator. The full WSJT decoder (version 4.9.5) was run

on each of the 11,000 simulated transmissions. The filled circles and solid curve in Figure 4 illustrate results from the Koetter-Vardy decoder. The essential conclusion is that 96% of the transmissions were decoded correctly at –23 dB, 41% at –24 dB, and 3% at –25 dB. No false decodes were produced by the KV decoder in any of the tests.

For the deep search algorithm, the filled squares and long-dashed curve show that 92% of the transmissions were decoded correctly at –27 dB, 58% at –28 dB, and 17% at –29 dB. Three "hard errors" (false decodes not flagged with a question mark) were recorded in the 11,000 simulated transmissions, for an overall error rate of 2.7×10^{-4} (too small to be seen in Fig. 4). If one includes decoded messages flagged with a question mark, the numbers for correct copy increase to 96%, 73%, and 29% at signal levels –27, –28, and –29 dB (short-dash curve and filled triangles). The error rate, illustrated by the short-dash curve at the lower left of Figure 4, reaches a maximum of 3.6% at –29 dB. With WSJT's "Aggressive decoding" option selected, the percentages of correct copy increase to 97%, 82%, and 41%, at –27, –28, and –29 dB (dotted curve and open triangles). However, the rate of false decodes also increases substantially, especially at –28 dB and below, reaching a maximum of 29% at –29 dB.

Similar measurements have been made for sub-modes JT65A and JT65C. The results are qualitatively similar to those shown for JT65B in Figure 4; the curves for JT65A are shifted about 1 dB to the left (more sensitive than JT65B), while those for JT65C are shifted about 1 dB to the right.

Normal JT65 messages cannot be decoded unless the sync vector is reliably detected. In WSJT the synchronizing procedure is exactly the same for sub-modes JT65A, B, and C. For the tests illustrated in Figure 4 with SNR less than about –29 dB, failure to synchronize is the cause of many failures to decode. Synchronization is very important for another reason, as well: correct synchronization may allow the decoding of an accumulated average message, independent of whether the transmitted message is decodable with the deep search algorithm. Measured rates of synchronization are illustrated in Figure 5, again using 1000 simulated transmissions at each value of SNR over a 10 dB range. Synchronization was achieved for 93% of the test transmissions at –28 dB, 74% at –29 dB, 44% at –30%, and 19% at –31 dB. These measurements imply that message averaging should typically succeed after about 3 transmissions at –26 dB and 8 transmissions at –28 dB, but will require as many as 20 transmissions at –29 dB. These conclusions are consistent with on-the-air experience with WSJT.

The simulator was also used to measure the detection rates for JT65 shorthand messages, as illustrated in Figure 5. With 1000 trials at each SNR, shorthand messages were correctly decoded in 88% of the trials at –31 dB, 60% at –32 dB, and 26% at –33 dB. The total number of incorrectly decoded shorthand messages was five, in 11,000 trials. All five would have been recognized as spurious by an attentive operator, because the measured frequency offset was much larger than the normal tolerances used.

For any of a large number of reasons, on-the-air performance of JT65 may differ somewhat from the simulated results shown here. The measurements summarized in Figures 4 and 5 were made under idealized conditions with additive white Gaussian noise (AWGN) and no fading. (An additional set of simulations has been made with the effects of Rayleigh fading included; the results are qualitatively similar to those shown here, with the curves shifted several dB to the right.) The effects of birdies, other interference, and non-Gaussian noise are harder to quantify. Suffice it to say that I often leave WSJT running in "Monitor" mode

for days at a time, with my receiver tuned to an arbitrary frequency between 144.100 and 144.160. I live in a densely populated region where plenty of birdies as well as other signals come and go on the 2 meter band. The typical rate of false decodes when monitoring a quiet band averages no more than one or two per hour. Examination of the files producing the spurious decodes nearly always reveals tell-tale evidence that would have caused an operator to recognize and reject the invalid information. When used in the intended manner, JT65 is a highly accurate communication protocol.

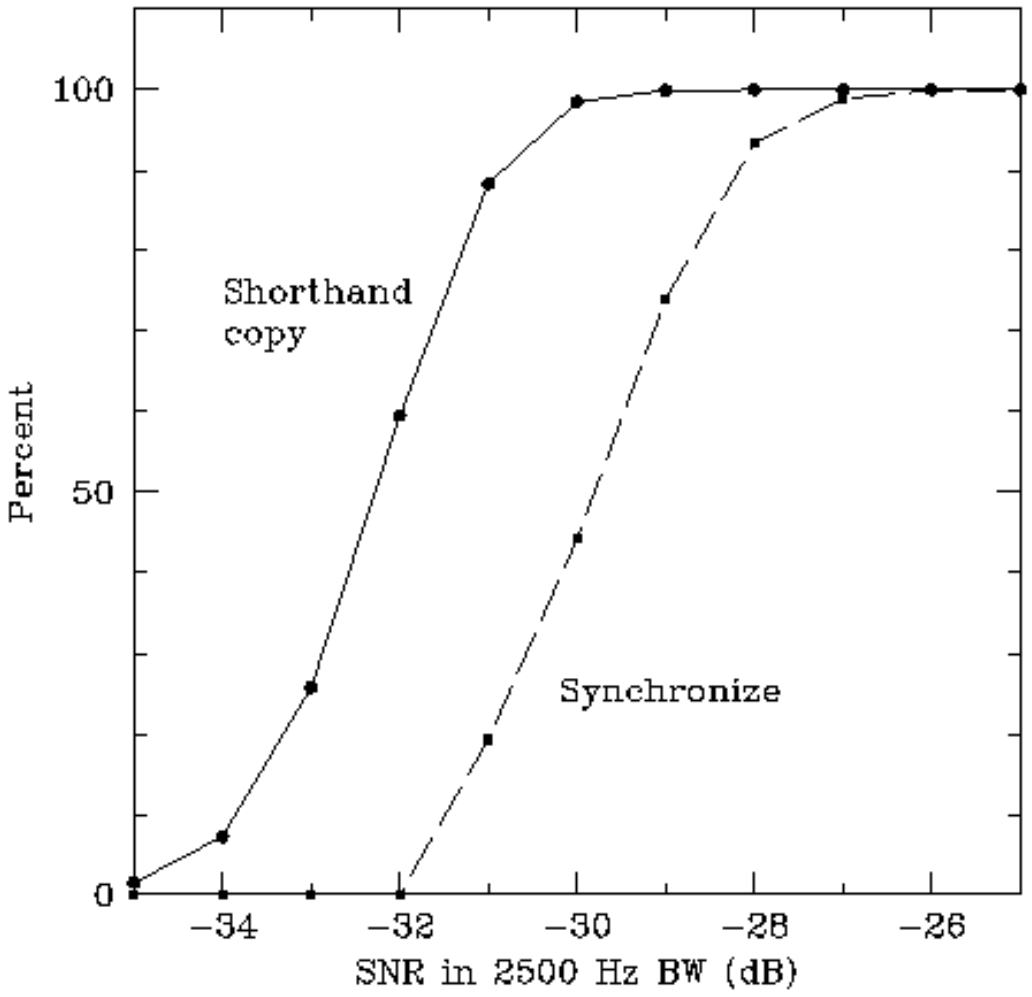


Fig. 5.– Rates of message synchronization and copy of shorthand messages as a function of SNR.

Version: March 9, 2005

JASON Version 0.99

Introduction

First of all, why the name Jason ? Well, you all know the program Argo...

Argo was the name of the mythological ship that brought the Argonauts in Colchis, searching for the Golden Fleece. The coxswain of the ship Argo was the Greek hero Jason. So this program, which loosely relies on the technology of Argo, has been named Jason.

And, if you don't like mythology, you can always read Jason as an acronym : **J**ust **A**nother **S**hip-**O**wner **N**ame :-)

The JASON Coding Scheme

The coding scheme of Jason is based on the ideas about IFK (Incremental Frequency Keying) initially proposed by Steve Olney, VK2ZTO.

Basically, the information is coded in the absolute value of the difference between two frequencies sent sequentially. This has the advantage of not needing a precise initial tuning. A tuning error of a few Hertz is perfectly acceptable.

Another characteristic is that, being the frequencies sent one at a time, you don't need a linear amplifier. A class-D Mosfet TX will do nicely. The frequency deltas can assume one of 16 different values. After sending one tone, the next one is shifted by the appropriate amount, up or down depending on the setting of the USB/LSB switch. With 16 deltas we need 17 slots (tones), and any overflow causes a wraparound.

With 16 possible deltas, each baud (a baud is a change in the signal transmitted, in this case a change in frequency) encodes 4 bits (called a nibble), which are not enough for a reasonable alphabet. So we need two nibbles for our alphabet.

But now a problem arises: how can we get character synchronization? In other words, which is the high-order and which is the low-order nibble? I solved the problem in the simplest way, which may be not the optimal one. I used the high order bit of each nibble to encode this info. So the high-order nibble is of the form '1xxx'b, while '0xxx'b is the low-order one. Here xxx stands for the actual information transmitted.

So now there are 6 bits available to encode our alphabet, enough for 64 symbols. I decided that my alphabet would be that that in ASCII code goes from x'20' (the blank) to x'5f'. This allows the transmission of all the letters (upper case only...), the ten digits, and practically all the punctuation symbols normally used.

JASON Signals

Ok, this explains the encoding. What about the signaling? For this version of Jason (but it may change in future versions), I chose the following parameters for the standard speed setting:

- The default Rx and Tx center frequency is 800 Hz.
- Each one of the 17 slots is separated from the adjacent by 3 FFT bins, so to guarantee some orthogonality.

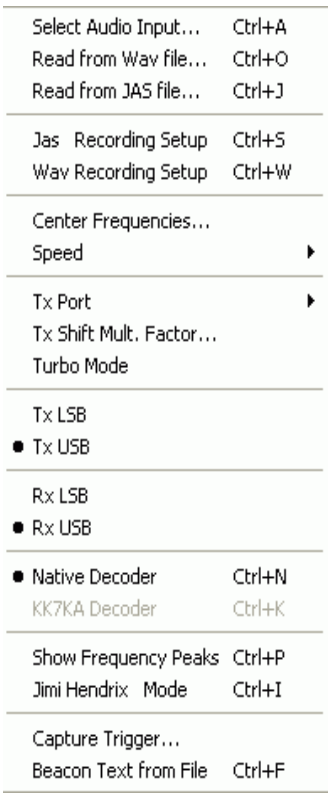
- Each FFT bin is roughly 84 millihertz for the standard ***Normal*** speed setting, so the slots are at a distance of roughly 252 millihertz, for a total band occupancy of 4.038 Hz for the standard ***Normal*** speed setting.
- Each baud (each tone sent) lasts for roughly 11.89 seconds for the standard ***Normal*** speed setting (the inverse of the FFT bin width)
- The throughput hence is about 2.5 characters per minute for the standard ***Normal*** speed setting, hardly a speed champion, but it compares favorably with QRSS.

The following table summarises the parameters for the various speed and turbo on/off settings.

Speed	Tone Duration(S)	Tone Spacing(Hz)	Bandwidth (Hz)	Characters/Minute (Turbo Off)	Characters/Minute (Turbo On)
Slow	95.2	0.03	0.5	0.3	0.6
Normal	11.9	0.25	4	2.5	5
Fast	1.5	2	32	20	40

Options

The ***Options*** menu allows settings for a number of parameters.



==========

- **Select Audio Input…:** Displays a Volume Control Dialog for your Soundcard Device. Here you can select the audio source input.
- **Read from Wav file…:** Displays an open file dialog to allow audio data to be read from a .WAV file (NOTE: Must have a sampling rate of 11025Hz).
- **Read from JAS file…:** Displays an open file dialog to allow data to be read from a .JAS file. A JAS file contains the down-sampled I and Q components of the audio input.

==========

- **Jas Recording Setup:** Displays a save file dialog allowing the specification of the .JAS filename and save location.
- **Wav Recording Setup:** Displays a save file dialog allowing the specification of the .WAV filename and save location.

==========

- **Center Frequencies…:** Displays a dialog that allows setting of the audio TX and RX frequencies (can be set independently) in the range of 50Hz – 5000Hz (the upper limit is set by the fixed 11025Hz sampling rate – Nyquist). Should initially be set to the default of 800Hz to allow easy netting.
- **Speed ->:** Displays a sub-menu to allow selection of three different speeds as shown in the table above (Slow/Normal/Fast). This setting must be the same at each end of the communication path.

==========

- **Tx Port ->:** displays a sub-menu allowing the selection of the three output options as outlined in the ***Interfacing JASON*** section below.
- **Tx Shift Mult. Factor…:** In the case where the output audio is fed to a frequency multiplying circuit (e.g., PLL) you need to enter the frequency multiplying factor here to allow JASON to reduce the frequency shift

deltas to ensure that the correct shifts are present at the multiplied frequency.

- **Turbo Mode:** An ON/OFF selection to enable/disable the TURBO mode. Selecting Turbo mode causes the tone time to be half as long while still maintaining the same tone shifts. This should only be done when the S/N is good. This setting must be the same at each end of the communication path.

  ==========

- **Tx LSB / Tx USB:** Selection must match the correct sideband setting for the Tx at your end.
- **Rx LSB / Rx USB:** Selection must match the correct sideband setting for the Rx at your end.

  ==========

- **Native Decoder:** Selects original decoder.
- KK7KA Decoder: Selects KK7KA decoder. Not selectable (greyed out) in v0.99 as decoder has not been upgraded by KK7KA to accommodate speed settings other than Normal.

  ==========

- **Show Frequency Peaks:** Enables/disables a real-time readout of the detected peak frequency between the yellow lines on receive.
- **Jimi Hendrix Mode:** Amplitude clipper intended to be used in the presence of impulsive noise. For those not atune to Mr. Hendrix – he was famous for a guitar sound produced by heavy clipping.

  ==========

- **Capture Trigger…:** Displays a dialog that allows entry of a piece of text that will trigger the beginning of text capture. Useful for leaving the receiver running waiting for a signal to begin (or rise out of the noise). The received screen text is saved in a file called “Jason.log” in the directory where Jason.exe is located.
- **Beacon Text from File:** Type different beacon messages into separate text files (.txt) and select a message from this open file dialog.

  ==========

Capture Range

The program has a capture range of 1.5 times the bandwidth (my arbitrary choice), i.e. slightly more than 6 Hz for the standard ***Normal*** speed setting. The capture range can be easily positioned with the mouse. To do this, left-click on the approximate center of the white lines that represent the received signal. The capture window (the two yellow lines)will then positioned so that its center will coincide with the frequency you have clicked on. Be warned that any signal that falls, even slightly, outside the capture window will be ignored by the decoding algorithm.

Frequency Stability Requirements

Using the standard ***Normal*** speed setting as an example, it is evident that the combination of the Tx and Rx drifts must be such that, in each 11.89 sec. interval, the frequency must stay in one single FFT bin, i.e. 84 millihertz. Actually, a drift from -1 to +1 bin is tolerated, and accounted for, by the program. But let's remain on the conservative side. If we translate this into short term stability, where short term means a period of 10 minutes, we compute that our oscillator must not drift, in each 10-minute interval, more than 0.084 * 600 / 11.89 = 4.24 Hz (roughly). This means, at 136kHz, a stability of 31 ppm is required, which doesn't look like a difficult figure to achieve.

Interfacing JASON

How does Jason interface with the radio?

For reception, it's quite easy. Just bring the Rx audio into the sound card, just as you do now with Argo and Spectran. Keep the audio level low. Jason works best when the level bar on the left side of the panel is below 50 %.

You will see in the waterfall window two yellow lines. Tuning must be done so that to ensure that the white lines of the received signal are always inside the two yellow lines. The program discards all that falls outside. Fine-tuning is possible with the mouse. Click on a signal line on the waterfall window, and its frequency will be brought at the center of the yellow lines (the receiving window). The Rx center frequency can also be adjusted with a menu choice.

For transmission, I have envisaged three different modes, to make interfacing the easiest possible.

1. **Parallel Port Output** : via the Options menu, you can choose between LPT1 or LPT2. The code (ranging from 0 to 16) for the tone to be sent is output, with the ***Strobe*** pin pulsed high for 100ms. The Tx condition is indicated by the ***SelectInput*** pin being high.

2. **Serial Port Output** : via the Options menu, you can choose between COM1, COM2 or COM3. The serial format is 9600 - 8N1. The code (ranging from 0 to 16) for the tone to be sent is output. The Tx condition is indicated by the RTS being active. Via the Option menu, you can also choose the ZL1BPU format, which is what is needed by Murray's AVR Atmel board (http://www.qsl.net/zl1bpu/MICRO/EXCITER) Basically, an ASCII 'T' is sent at the beginning of transmission and an 'X' at the end. Moreover, the tones to be sent are identified by the three character sequence 'A00', 'A01'...'A09', 'A0A', A0B'....'A0F', 'A10' . No CR/LF used.

3. **Audio Output** : selectable via the Options menu.
   If you choose this method, a note is generated using the sound card, with a software implementation of a DDS, with a very long sine table (262144 entries), which ensures low distortion. The default center frequency generated is 800 Hz, but can be selected via the ***Options*** menu in the range 50Hz to 5000Hz.

   When using audio output, there is the possibility to specify a shift multiplier factor (default = 1), which can be handy when generating the RF using a PLL process that involves divisions. The output is in stereo with left and right channels outputting the audio in quadrature (I and Q) allowing feeding to a phasing-type SSB transmitter set-up. To maximise sideband rejection of a phasing-type SSB transmitter a dialog for adjusting the relative amplitude and phase of the two stereo channels (I and Q) is available in the ***Options | Tx Port*** menu.

With these interfacing possibilities, it should be easy to hook-up a Tx to Jason, both if you have the capability to up-convert the audio tone to the working frequency, or if you use a DDS board, either with a parallel or serial interface.

If you use a DDS board, you will need the following table to set up the frequency to generate for the standard speed setting, depending on the code output from Jason. Add the value in the table to the nominal value of the carrier generated by the DDS.

Tone #	Frequency (Hz)
0	797.981
1	798.234
2	798.486
3	798.738
4	798.991
5	799.243
6	799.496
7	799.748
8	800.000
9	800.252
10	800.505
11	800.757
12	801.009
13	801.262
14	801.514
15	801.766
16	802.019

These are also the default frequencies for the standard ***Normal*** speed setting generated when using the audio output, provided that your sound card has an exact sampling rate.

The encoding is taken care of by Jason; the DDS board task is only to generate the appropriate frequency, according to the code received via the serial or parallel port.

Frequency Accuracy for Using Audio Output Interface

A common method for interfacing is using a soundcard that is normally installed in a PC as standard. It is useful to analyse the frequency accuracy and stability requirements for JASON. JASON assumes an exact 11025Hz sample rate. The initial frequency accuracy determines the ability to place the desired signal in the input frequency range of JASON. The table below outlines requirements for different speeds.

	Bin Width (Hz)	Display Range (Hz)	Capture Range (Hz)	Required Capture Window Accuracy (Hz)	Required Display Window Accuracy (Hz)
Slow	0.0105	2.69	0.76	**±0.1**	**±1**
Normal	0.0841	21.5	6.06	**±1**	**±9**
Fast	0.673	172.3	48.45	**±8**	**±70**

The final frequency accuracy is the sum of the receiver frequency error and the soundcard error. Sampling frequency accuracies of soundcard are usually much poorer in terms of ppm than modern receivers. Typically soundcards sampling frequencies are derived from a standard ±100ppm crystal oscillator while receivers are typically ±5ppm. Working in favour of the soundcard is that it is dealing with audio frequencies; therefore 100ppm @ 800Hz = 0.08Hz error. For 5ppm accuracy receiver working at 134kHz the error is about 0.7Hz. So worse error is 0.78Hz (say 0.8Hz). Short-term drift of both the soundcard and receiver will be much less than the initial accuracy error. Therefore with this combination it is possible to start JASON in the ***Fast*** mode without having to manually adjust the signal to be within the capture window (assuming both ends of the communication path are using gear of the same specifications). If

the operators are prepared to manually adjust the capture window then the signal need only be within the display window requiring 8 times less initial accuracy and so the ***Normal*** mode is also possible without further calibration for the equipment accuracy described above.

Important Disclaimer: Older soundcards usually had ±100ppm accuracy for the standard sampling rates of 44100, 22050, 11025 samples/second. However, the newer soundcards that have 96kHz and 48kHz sampling rates will likely have ±100ppm for these two rates, but the standard rates of 44100, 22050, 11025 might not have the same accuracy. For example a newer model external soundcard might be off by about 64ppm for the 48kHz and 96kHz sampling rates, while the sampling rates for 44100, 22050 and 11025 might be off by many ppm!!! Because of this it is probably necessary to calibrate the soundcard if it is one of the newer cards.

Methods for calibrating soundcards can be found on the Internet. As JASON has no provision to enter the real sampling rate (it assumes exactly 11025Hz) you can compensate by changing the ***Rx*** centre frequency. For example, if you find your soundcard sampling frequency is high by 100ppm (i.e., 11026.103Hz), you should enter a ***Rx*** centre frequency 100ppm low (e.g., 799.92001Hz for a nominal 800Hz). Of course, this does not account for any receiver errors.

WAV and JAS Recording

Incoming signal data can be saved for reading back at a later date. Two formats are available – WAV and JAS. The WAV format is Mono, 11025Hz sample rate, 16-bits per sample. The JAS format brings out the internal down converted and down sampled I/Q data (much lower effective sampling rate) that is normally fed to the internal JASON complex FFT algorithm before decoding.

Starting and stopping WAV and JAS recording is done via the 'W' and 'J' buttons respectively that appear just to the right of the ***Options*** and ***About*** buttons when JASON is in Rx mode. Note that if you haven't setup the filenames for saving to, you will be presented with a file save dialog when you first click these buttons. So if you want to suddenly capture some interesting signals make sure you have setup the filenames beforehand (See WAV and JAS recording setup in the ***Options*** menu).

After recording a WAV or JAS file you can feed it back later to JASON via the WAV and JAS reading modes in the ***Options*** menu.

The purpose of recording to a JAS file is that, compared to a WAV file, the size of a JAS file at ***Slow, Normal*** and ***Fast*** speeds is 0.4%, 3% and 25% respectively of a WAV file for the same record time. Handy for recording overnight sessions without filling up your hard drive!!! Of course there is a catch. The JAS file only records the signal passing into the internal complex FFT and decoder – therefore you cannot change the centre frequency or speed if you discover that the signal is outside the capture range or at the wrong speed when you play the JAS file back later. So if you see a nice signal but some of it or all of it falls outside the capture and/or if it is the wrong speed then too bad. Any changes you make are ignored during playback. In comparison the WAV file records the whole audio signal and when played back allows adjustment of centre frequency and speed. So if you see a nice signal that is wholly or partly outside the capture window and/or the wrong speed, you can make changes via the ***Options*** menu and run the WAV again. The playback of the WAV file is just like a speeded up version of real-time operation (so you have to be quick if you want to click in the display window to centre the signal unless the file is long :-).

JASON Beaconing

For weak signal work it is useful to be able to repeat a message continuously to allow other stations to wait for propagation conditions to be suitable for reception. This can be done by typing in the text message to be repeated in the typing input window enclosed in curly braces like this { *text to beacon* }. For example { I2PHD IN JASON MODE }. Alternatively, a text file can be created containing the beacon text (say in Notepad) and the file selected from the ***Options*** menu.

I do hope this notes are sufficient to make you understand how Jason works, and how to interface it with a Tx for LF work.

Should you have any questions, feel free to contact me at dibene@usa.net.

Enjoy,

Alberto I2PHD

By Murray Greenman, ZL1BPU

MFSK for the New Millennium

Following in the wide wake of digital-mode revolutionary PSK31, MFSK (in its myriad flavors) is raising the digital performance bar ever higher. And, like PSK31, all you need to get into the action is a computer, a sound card and a free download. Try the newest super-RTTY for yourself!

Over the past two years, digi-mode DXers have had access to "designer" digital modes that offer greatly improved HF performance when compared to "classic" RTTY. The best-known "RTTY replacement" is probably PSK31, which is a great DX performer with high and low power alike. Several landmark PC programs for PSK31 are now available and have recently been reviewed in many Amateur Radio magazines.[1,2]

Unfortunately, none of these digital modes—PSK31 included—has been able to counter *all* of the problems prevalent on HF. The list of troubling phenomena includes multi-path reception, Doppler flutter and severe lightning and man-made noise (common on 160 to 40 meters). To counter these effects, some hams have resorted to "fuzzy" image modes such as Feld-Hell and PSK-Hell.[3] I can now report that major progress has been made in solving *all* of these problems with a true digital mode.

The new mode is MFSK16, and it uses techniques from the '60s and the latest advances in digital signal processing to provide truly remarkable results. MFSK16 won't replace PSK31 completely, although it provides a viable alternative when other modes won't get through. It could replace RTTY, however, and by the time you read this you may have already heard its distinctive sound on the bands.

History

The first multi-tone data mode wasn't digital—it was "fuzzy." The LMT Seven Tone Radio Mode[4] dates from 1937 and was used to portray text as images, rather like Hellschreiber. The best known examples of digital MFSK (Multi-tone Frequency-Shift Keyed) modes are Piccolo[5] and Coquelet[6], which both date from the early 1960s.

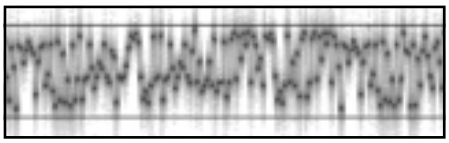


Figure 1—An MFSK16 spectrogram (the horizontal lines are 300 Hz apart).

MFSK is really a type of super-RTTY, and it's difficult to understand why hams didn't adopt it (or adapt it) years ago. The MFSK technique was developed during the heyday of HF teleprinter communications as a way to combat multi-path propagation problems and provide reliable point-to-point communications with relatively simple equipment. Piccolo, for example, was used on diplomatic links between England and Singapore, and typically provided good copy for an hour after RTTY links had faded out. The technology was then electromechanical, but several key principles were recognized and exploited at the time:

- Performance (reduced error rate) improved as the number of tones used increased.
- Performance was best when the least number of symbols[7] was used to represent each transmitted text element.[8]
- In systems that used special integrating detectors, tones spaced as closely as the baud rate could be uniquely detected without cross-talk.

Piccolo and Coquelet both used two symbols per text character—compared to 7.5 for RTTY and 3 to 12 for PSK31. MFSK16 uses only *one* symbol per signaling element! With MFSK, the baud rate (the rate at which symbols are transmitted) is quite a bit lower than the text rate because each symbol carries more information in its frequency properties than RTTY or PSK. Although it is somewhat confusing, this technique has an advantage because "longer" symbols are easier to detect in the presence of noise, they have a narrower bandwidth and are much less affected by multi-path timing errors.

Piccolo originally used as many as 32 tones, but the most common form used six. Coquelet generally used 12 tones. MFSK has recently been tested with as many as 64 tones, although the released version, MFSK16, uses 16 tones and the weak-signal variant, MFSK8, uses 32.

The integrating detector used in Piccolo was a milestone in FSK detection techniques in its day.[9] Without going into great detail, narrow active filters with very high gain were used to detect each tone. By carefully choosing the baud rate and tone-channel spacing and resetting the filters at the start of each symbol period it was possible to reliably detect very weak tones without cross-talk. In fact, the response of the adjacent channels produced a null at the sampling point. This helped with noise rejection and prevented energy resulting from ionospheric effects on one tone from appearing in the next channel.

With the advent of satellite communications and high-speed ALE (Automatic Link Establishment) systems, these older commercial MFSK modes have largely fallen into disuse. The concepts and the

[1]Notes appear on page 36.

technology are still viable, however, and should be of great interest to radio amateurs faced with the age-old problems of multi-path, Doppler instability and interference.

The New Approach

In searching for a better way to hold reliable long-path QSOs, I looked at what made copy difficult with existing modes and what could be done about it. It was obvious that phase-shift keying (PSK), unless relatively high speed, wasn't practical. The incidental phase errors introduced by an unstable ionosphere (particularly in polar regions) typically exceed the phase modulation of the signal. Frequency-shift keying (FSK) and on-off keying also perform poorly, but principally because the arrival time of signals vary, often by as much as five to 10 ms, depending on the path, and perhaps by as much as 30 ms between long and short paths. This interval is longer than the signaling duration of a 22-ms RTTY symbol—and multi-path reception is the reason why so many RTTY signals, even strong signals, don't print reliably.

While casting around for a better method, I revisited the MFSK techniques mentioned previously. At the same time I also reviewed the advances made in modern PC and sound card DSP technology, which were light years ahead of 1960's hardware, especially in compactness and simplicity. Putting these together, I had all the necessary building blocks to replicate and enhance the old MFSK modes using nothing more than a PC with a sound card!

I decided to kick things off by sending a specification for the new mode to a bunch of DSP, coding and software experts, and a remarkable collection of ideas and offers of assistance resulted. Nino Porcino, IZ8BLY, of Hellschreiber and MT63 fame, quickly turned the specification into reality. The result has been tested thoroughly in real and simulated conditions. The first QSO using this new mode (between Nino and myself) was over an 11,000-mile long-path connection on 17 meters. We had 100% copy using 25 W and dipole antennas—so the specification can't be too far off the mark!

That amazing day was June 18, 2000, and since then we have been in communication using this mode almost every day. Most days we work 20 or 17 meters using as little as 5 W.

As with the popular PSK31, all you need to run this amazing new mode is a Pentium-class PC with a sound card and a couple of simple cables. You could even use it with the QRP transceiver designed by Dave, NN1G.[10] The first software for MFSK16 is called *Stream* and is available from the MFSK Web site[11] or from IZ8BLY.[12]

Figure 2—*Stream* in the MFSK16 mode.

The Signal

What does this new mode consist of? Well, there are 16 tones—sent one at a time—at 15.625 baud and spaced 15.625-Hz apart. Each tone represents four binary data bits. The transmission is 316-Hz wide and has a ITU-R specification of *316HJ2B*.[13] It's exactly like RTTY, but with 16 closely spaced tones instead of two wider-spaced tones. With a bandwidth of 316 Hz, the signal easily fits through a narrow CW filter.

The tones are continuous phase keyed, which eliminates keying noise, and the phase information can be used to determine tuning and symbol phase. Figure 1 shows an MFSK16 spectrogram (the horizontal lines are 300-Hz apart).

Unlike Piccolo or PSK31, no special arrangements are made to transmit symbol timing, which can be recovered from the inherent properties of the signal. One critical factor, like RTTY, is that the signal is of constant amplitude and does *not* require a linear transmitter to maintain signal purity.[14] Unlike SSB and PSK31, overdriving the transmitter will *not* make an MFSK16 signal any wider.

To ensure that text is received with an absolute minimum of errors, the new mode incorporates an excellent forward error-correction (FEC) technique using *Viterbi* decoder routines developed by Phil Karn, KA9Q, and a clever self-synchronizing interleaver developed for MFSK by IZ8BLY. The typing rate, even with FEC, tops 40 WPM. This speed is achieved by efficient coding techniques, including a varicode similar to PSK31, which provides an extended ASCII character set.

Finally, the receiver detector uses a synchronous Fast-Fourier-Transform (FFT) routine, a DSP technique that exactly models the original Piccolo integrating detector. The FFT also provides phase information, automatic frequency control (AFC) and a "waterfall" tuning display. The filter provides 4-Hz channels and is easily able to separate the 16 closely spaced tones.

The signal has an amusing musical sound, is quite narrow, is clean to tune across and not unpleasant to listen to. The sound is certainly better and the bandwidth narrower than many HF modes in use today.

First Impressions

Downloading the software and installing it is very simple. The "help file" is also available as separate download, so you can read that before you install the software. Figure 2 shows *Stream* in MFSK16 mode.

At first glance the software is well laid out and similar in appearance to *IZ8BLY Hellschreiber* or *MT63*, which isn't surprising, considering its origin. It has a generous collection of tools along the top of the screen, separate transmit and receive windows, a good collection of definable "macro" buttons and an excellent "waterfall" tuning display. Along the bottom is a list of settings and parameters, plus the date and time. There is also a drop-down log window for automatic logging and insertion of QSO information and a useful "QSP" window for relaying incoming text.

Nino's software actually includes *three* new modes! The default mode is MFSK16 (16-tone, 16-baud MFSK with FEC). Next is a slower, but more sensitive, variant called MFSK8 (32-tone, 8-baud MFSK with FEC). Both modes share the same 300-Hz bandwidth, but sound quite different. The other new mode is Nino's PSK63F, which is a 63-baud PSK mode that's similar to PSK31, but faster and with full-time FEC. PSK63F has about a 100-Hz bandwidth.

The MFSK and PSK modes are complementary, as Nino's new mode is great for short-path DX and local QSOs. You'll have no trouble telling them apart, and no trouble telling Nino's PSK63F from PSK31 because it is twice as wide. As a standard of comparison [and perhaps the ultimate in convenience—*Ed.*], the software includes PSK31 as well!

Stream is quite simple to use—start typing and it transmits; press F12 to end the transmission. The challenge comes in getting a signal lock. It takes some skill and a certain amount of patience to learn how to properly and efficiently tune an MFSK signal. I'm confident you'll agree that the results are worth the effort.

Because the tones are closely spaced and the filters quite narrow, you *must* have a stable transceiver and you *must* use software tuning—not transceiver tuning and certainly not the RIT! The software tunes up and down in 1-Hz steps. Click on the waterfall, with its zoom

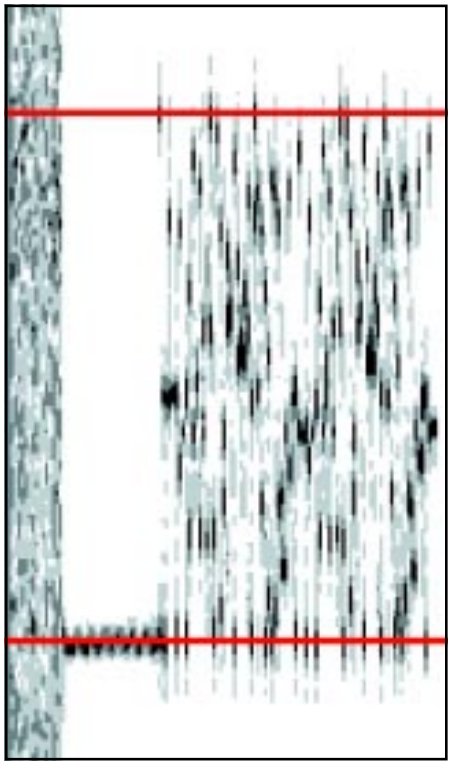


Figure 3—The *Stream* waterfall display in 3X zoom

function, for exact tuning.

The software's AFC is good, but you need to be within about 5 Hz of dead center to get a good response. The AFC works on the idle tone, which appears at the start of every over and also during transmission. Whenever the AFC is active, the Phase Scope comes alive. You can also manually tune by clicking on the waterfall display in just the right spot or by using the Up/Down frequency buttons to tweak the tuning.

There's an interesting display alongside the Phase Scope that shows the Symbol Clock Alignment. This display is a great indication of ionospheric stability! The Bit Shape display is a small oscilloscope that shows what the symbol sync is working with (this is a type of correlator).

Tuning is done using an excellent waterfall display. Figure 3 shows this display in 3X zoom mode. Under the lower horizontal line (red on the screen) you'll see a broad band towards the left. This is the idle carrier, the lowest of the 16 tones. This carrier is transmitted briefly at the start of each over and returns at the end, or whenever the operator stops to think. To tune the signal correctly, center the red line on this carrier and the AFC will keep it there. During the over you'll see little black vertical stripes all over the waterfall, with gray "side-lobes" above and below. These are the transmitted symbols, and once again, you can adjust the software tuning so the red line centers on the lowest of these symbols. Unfortunately, while this is easy when the signal is already tuned, finding the correct spot on a weak signal during a transmission isn't so simple and takes a little practice.

Once you've found the right spot, almost-perfect text will start to appear on the screen, delayed by three to four seconds as the data trickles through the error-correction system and appears one or two words at a time. You'll soon get used to that.

The mode is a delight to use once you learn to tune it. The typing speed is fast and, while transmit-receive changeover isn't as fast as RTTY or Hellschreiber, it's fine for conversing and net operations.

Performance

Well, this *is* the telling factor, isn't it? For short-path QSOs out to 8000 miles (with no polar propagation), MFSK16 works fine but you may find PSK31 easier to use. If you're interested in QRP, MFSK16 appears to be the hands-down winner. Over long-path and polar routes—and when conditions are really nasty—MFSK16 stands alone. It keeps giving almost perfect copy when signals are barely audible, have bad fades, noticeable Doppler, multi-path distortion and even QRM. High power *isn't* necessary.

MFSK16 is also probably the best mode yet for digital work on the lower bands. If you are into traffic handling or sending bulletins on 80 or 40 meters, give this mode a try. It just doesn't give up! 80 meters is especially prone to multipath, as RTTY and Feld-Hell users know. On nighttime 80-meter QRP circuits, MFSK16 will work over thousands of miles with 90% perfect copy! As a bonus, lightning effects are largely ignored.

Although not noticeably better on the low bands, MFSK8 is great to have when the band starts to die. It's definitely more sensitive than MFSK16 and, although tuning is very tight and typing speeds are down to 25 WPM, it will allow you to complete that difficult QSO with almost perfect copy.

PSK63F, on the other hand, is quite the reverse. Although not very good on long path, it's sensitive (almost as good as PSK31) and fast (40 WPM). Thanks to FEC, it provides error-free copy most of the time. It's also very easy to tune, as it's wider than PSK31 and has excellent AFC performance. PSK63F is also minimally affected by Doppler and drift problems. It's good for short-haul DX and would be great on VHF.

Stream has been tested on an ionospheric simulator by Johan Forrer, KC7WW, and the results bear out the practical experience. Moe Wheatley, AE4JY, has run sensitivity tests that place MFSK16 on an equal footing with PSK31 in white noise. As testing proceeds, speed and tone tweaks may produce improved performance or even new modes.

Feedback from users shows that sensitivity and the ability to cope with poor conditions are unsurpassed. MFSK16 is also useful on VHF—it's not affected by aircraft reflections and is great for DX because of its inherent sensitivity. Karl Schneidhoffer, HA5CAR, is even using it on 23 cm. Here are some quotes from users:

"Great—just great!"—*Iván, LU3OK*

"Bob, K4CY, appeared out of the noise. A good QSO followed. Band propagation not ideal at this time"—*Victor, G3GK*

"It's got one strong receiver/decoder, doesn't it? I was getting 70%+ copy on signals that wouldn't even change the shade of gray on the waterfall!" —*Gordon, N5AJF*

"Hey guys—great mode! Works well down into the noise for me. Andy (KBØEOQ) and I went to the 5-W area for a bit last night and I still pulled him in when QSB took him below my QRN level and audibility"—*Gary, AGØN*

"Conditions were slow QSB and aircraft reflections. Tuning stability not too much of a problem. Signals were about

Is it Legal?

MFSK16 uses publicly available software, and the source code, algorithms and codes are all public domain.[15]

In that sense the mode is open, unencrypted and publicly available. In fact, users with the ability could write their own versions. I hope that other versions will appear, including a *Linux* version.

According to Chris Imlay, W3KD, ARRL General Counsel,[16] the legality of new modes in the USA should be determined by reviewing FCC Part 97, section 97.3(c), to determine which emission type applies (based on the emission designator, in this case F1B). This is the same emission designator as RTTY. According to both Chris Imlay and Paul Rinaldo, W4RI (ARRL Technical Relations Manager), MFSK16 meets the FCC requirements for a legal HF digital mode.

Of course, *where* to operate the new mode also depends on its emission type. MFSK16 calling frequencies on DX bands are 10.147 MHz USB (indicated dial frequency—that's 10.148 MHz idle carrier), 14.080 MHz USB, 18.105 MHz USB and 21.063 MHz USB.—*ZL1BPU*

S1 or below"—*Terry, GØEZY (100 mile path on 2 meters)*

"I worked RTTY since about 1978, but gave it away. Copy not very good except when signals are strong. It's the same with PSK—no good at all on really long DX contacts. This mode is much, much better."—*Frank, ZL2BR*

Try It Yourself!

At press time the only publicly released software available for MFSK16 is *Stream* by IZ8BLY. It's completely free and fully functional, and can be downloaded from numerous places on the internet.[17] *Stream* requires at least a Pentium 100 PC with a 16-bit sound card and *Windows 9x* or newer. Other versions will hopefully follow. You can subscribe to the MFSK16 support group by sending an e-mail to **MFSK-subscribe@egroups.com**.

If you're already set up for PSK31, you only need to download *Stream* and you're ready to go. Even if you've never tried sound-card-based digital communication software before, you'll be surprised at how easy it can be. The *Stream* help files will tell you how to connect the necessary audio and keying cables between your computer and your radio. You'll find these files in the "Help" folder that is created when you install *Stream*.

This month you'll have a chance to test the performance of MFSK16 during the W1AW HF Digital Run. See the announcement elsewhere in this issue for details.

Notes

[1]Steve Ford, WB8IMY, "PSK31 2000," *QST,* May 2000.

[2]Jack Heller, KB7NO, "Inside MIXW32," *73 Amateur Radio Today*, Jul 2000.

[3]Murray Greenman, ZL1BPU, "Let's See you in Hellschreiber," Jan 2000.

[4]See **www.qsl.net/zl1bpu/Fuzzy/LMT.html**.

[5]Developed by H. K. Robin and J. D. Ralphs, et al, for the British Foreign and Commonwealth Office.

[6]Developed in Belgium by ACEC and used by French and Belgian police.

[7]The smallest signaling entity of a digital mode.

[8]Article (in French) in *ACEC-Revue, No. 3.4*, 1970.

[9]H.K. Robin, OBE, et al, "Paper 204E," published in *Proc IEE, Vol 110, No. 9*, Sep 1963.

[10]Skip Teller, KH6TY, and Dave Benson, NN1G, "A Panoramic Transceiving System for PSK31," *QST,* Jun 2000.

[11]**www.qsl.net/zl1bpu/MFSK/**

[12]**space.tin.it/computer/aporcino**

[13]See FCC Part 47, paragraphs 2.201 and 2.202.

[14]Distortion in sound card or transmitter audio stages can still lead to undesirable images of the signal above and below the correct one.

[15]**www.qsl.net/zl1bpu/MFSK/Documents.html**

[16]See "Is Hellschreiber Permissible Under Part 97?" *QST,* Jan 2000, p 54.

[17]For example, from **www.qsl.net/zl1bpu/MFSK/software** or **www.egroups.com/files/MFSK**. At time of writing the latest version is *StreamSetup085.EXE*, **www.qsl.net/zl1bpu/MFSK/software/StreamSetup083.EXE**.

You can contact the author at 94 Sim Rd, Karaka, RD1, Papakura, New Zealand; **as149@detroit.freenet.org**.

NEW BOOKS

A FAMILY AFFAIR: THE R. L. DRAKE STORY

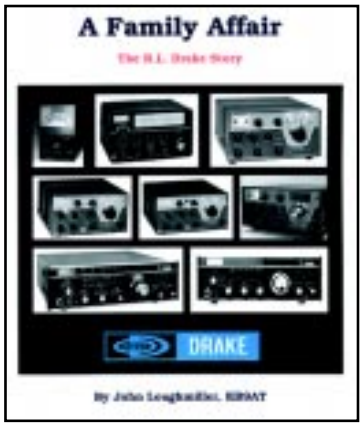
By John Loughmiller, KB9AT

Published by Technical Support Group, 15 Saddle Ridge Trail, Alexandria, KY 41001-9105; tel 859-635-6487; **home.fuse.net/tsg/**. *First edition, softcover, 305 pages, 8½ × 11 inches with black and white illustrations. $29.95 plus shipping and handling.*

Reviewed by Steve Ford, WB8IMY
QST *Managing Editor*

I must confess at the beginning that I bring a special bias to this review of *A Family Affair: The R. L. Drake Story*. I was an employee of the R. L. Drake Company in the mid-1980s. So, when I received a copy of the book by John Loughmiller, KB9AT, I approached it with some trepidation. Would Loughmiller accurately portray the Drake Company that I knew?

As it turns out, Loughmiller's chronicle of Drake's history is honest and on target. He has carefully researched his subject, spinning a fascinating story that rises above a mere collection of historical facts. At the heart of *A Family Affair: The R. L. Drake Story* are the recollections culled from a number of former employees who have since retired or otherwise moved on. At times the tone of the book is almost gossipy, but not in a negative sense. It treats the reader to inside glimpses of life at the company that would otherwise have vanished into the mists of time. You feel like the proverbial "fly on the wall," privy to private conversations and "unusual" episodes (failed designs, labor union bickering, interpersonal conflicts—it's all there). Because none of the juicier items come from my time with the company, I can't vouch for their accuracy first hand, but I knew the individuals involved and the stories certainly have a strong ring of truth.



I found only one of the anecdotes to be a little off target. On page 27 there is the story of a "Hint & Kink" item published in the February 1969 *QST* concerning a method for cleaning a badly contaminated radio by flushing it with water and baking out the residual moisture in an oven. A teenage TR-3 owner attempted this, but used too much heat. The results were disastrous. When the boy's mother complained to Bob Drake, he fixed the radio free of charge. According to the book, the advice was an early *QST* April Fool joke. In reality, it wasn't, but it is true that Bob Drake carried a grudge for *QST* for a long time thereafter.

Like many good historical works, *A Family Affair: The R. L. Drake Story* has a tragic rise-and-fall structure, beginning with the company's birth under the direction of Bob Drake, along with the birth of receivers and transmitters that were destined to become legendary. You witness the ascension of Drake's Amateur Radio line up to and including the development of the famous TR-7 transceiver. You also witness Drake's exit from the Amateur Radio stage as the rig that was to become the TR-8 is crated and sent to storage, never to exist beyond a prototype.

Loughmiller's writing is outstanding, a cut above most of the material you find in the Amateur Radio press. Thanks to his compelling narrative, *A Family Affair: The R. L. Drake Story* is quite a page-turner. The story pulls you along like a well-crafted fiction novel.

The Drake story itself comprises about half of the book. The remaining half is devoted to technical information about various Drake products. It is an invaluable encyclopedia of modifications, hints and tips.

A Family Affair: The R. L. Drake Story is a must-have for Drake aficionados, but it has an appeal that reaches beyond die-hard devotees. The story that unfolds in the book is a microcosm of the history of Amateur Radio itself. It is a somewhat cautionary tale that John Loughmiller has woven in an entertaining, informative style.

AUTOMATIC POSITION REPORTING SYSTEM

APRS PROTOCOL REFERENCE

Protocol Version 1.0

Authors	The APRS Working Group
Document Version	**Approved Version 1.0.1**
Filename	aprs101.pdf
Date of Issue	**29 August 2000**
Copyright	
Technical Editor	Ian Wade, G3NRW

APRS Protocol Reference
Protocol Version 1.0

by the APRS Working Group

Edited by Ian Wade

Published by
Tucson Amateur Packet Radio Corp
8987–309 East Tanque Verde Road, #337
Tucson
AZ 85749-9399
United States of America

http://www.tapr.org

ISBN 0-9644707-6-4

TAPR Publication Number: 99-4

FOREWORD

This APRS Protocol Reference document represents the coming-of-age of WB4APR's baby. Starting with a simple concept — a way to track the location of moving objects via packet radio — programs using the APRS protocol have grown into perhaps the most popular packet radio application in use today. It's also become one of the most complex; from the simple idea grew, and still grows, a tactical communications system of tremendous capability. Like many ham projects, the APRS protocol was designed as it was being implemented, and many of its intricacies have never been documented.

Until now. This specification defines the APRS on-air protocol with a precision and clarity that make it a model for future efforts. The work done by members of the APRS Working Group, as well as Technical Editor Ian Wade, G3NRW, should be recognized as a tremendous contribution to the packet radio art. With this document available, there is now no excuse for any developer to improperly implement the APRS protocol.

As an APRS Working Group member whose role was mainly that of observer, I was fascinated with the interplay among the APRS authors and the Technical Editor as the specification took form. Putting onto paper details that previously existed only in the minds of the authors exposed ambiguities, unconsidered consequences, and even errors in what the authors thought they knew. The discussion that followed each draft, and the questions Ian posed as he tried to wring out the uncertainties, gave everyone a better understanding of the protocol. I am sure that this process has already contributed to better interoperability among existing APRS applications.

Everyone who has watched the specification develop, from the initial mention in April 1999 until release of this Version 1.0 document in August 2000, knows that the process took much longer than was hoped. At the same time, they saw the draft transformed from a skeleton into a hefty book of over 110 pages. With the specification now in hand, I think we can all say the wait was worth it. Congratulations to the APRS Working Group and, in particular, to G3NRW, for a major contribution to the literature of packet radio.

John Ackermann, N8UR

TAPR Vice President and APRS Working Group Administrative Chair

August 2000

APRS PROTOCOL REFERENCE

TABLE OF CONTENTS

APPENDICES

PREAMBLE

APRS Working Group

The APRS Working Group is an unincorporated association whose members undertake to further the use and enhance the value of the APRS protocols by (a) publishing and maintaining a formal APRS Protocol Specification; (b) publishing validation tests and other tools to enable compliance with the Specification; (c) supporting an APRS Certification program; and (d) generally working to improve the capabilities of APRS within the amateur radio community.

Although the Working Group may receive support from TAPR and other organizations, it is an independent body and is not affiliated with any organization. The Group has no budget, collects no dues, and owns no assets.

The current members of the APRS Working Group are:

John Ackermann, N8UR	Administrative Chair & TAPR Representative
Bob Bruninga, WB4APR	Technical Chair, founder of APRS
Brent Hildebrand, KH2Z	Author of APRS+SA
Stan Horzepa, WA1LOU	Secretary
Mike Musick, N0QBF	Author of pocketAPRS
Keith Sproul, WU2Z	Co-Author of WinAPRS/MacAPRS/X-APRS
Mark Sproul, KB2ICI	Co-Author of WinAPRS/MacAPRS/X-APRS

Acknowledgements

This document is the result of contributions from many people. It includes much of the material produced by individual members of the Working Group.

In addition, the paper on the Mic-E data format by Alan Crosswell, N2YGK, and Ron Parsons, W5RKN was a useful starting point for explaining the complications of this format.

Document Version Number

Except for the very first public draft release of the APRS Protocol Reference, the document version number is a 3-part number “P.p.D” (for an approved document release) or a 4-part number “P.p.Dd” (for a draft release):

Document Version Number			
APRS Protocol Version		***Document Release***	***Draft***
Major Release	***Minor Release***		
P.	p.	D	d

Thus, for example:

- Document version number "1.2.3" refers to document release 3 covering APRS Protocol Version 1.2.
- Document version number "1.2.3c" is draft "c" of that document.

Release History

The release history for this document is listed in Appendix 7.

Document Conventions

This document uses the following conventions:

- `Courier font` ASCII characters in APRS data.
- ␣ ASCII space character.
- … (ellipsis) zero or more characters.
- `/$` Symbol from Primary Symbol Table.
- `\$` Symbol from Alternate Symbol Table.
- 0x hexadecimal (e.g. 0x1d).
- All callsigns are assumed to have SSID –0 unless otherwise specified.
- **Yellow marker** (appears as light gray background in hard copy). Marks text of interest — especially useful for highlighting single literal ASCII characters (e.g. **"**) where they appear in APRS data.
- Shaded areas in packet format diagrams are optional fields.

Feedback

Please address your feedback or other comments regarding this document to the TAPR *aprsspec* mail list.

To join the list, start at http://www.tapr.org and then follow the path Special Interest Groups ➲ APRS Specification ➲ Join APRS Spec Discussion List.

AUTHORS' FOREWORD

This reference document describes what is known as *APRS Protocol Version 1.0*, and is essentially a description of how APRS operates today.

It is intended primarily for the programmer who wishes to develop APRS-compliant applications, but will also be of interest to the ordinary user who wants to know more about what goes on "under the hood".

It is not intended, however, to be a dry-as-dust, pedantic, RFC-style programming specification, to be read and understood only by the Mr Spocks of this world. We have included many items of general information which, although strictly not part of the formal protocol description, provide a useful background on how APRS is actually used on the air, and how it is implemented in APRS software. We hope this will put APRS into perspective, will make the document more readable, and will not offend the purists too much.

It is important to realize how APRS originated, and to understand the design philosophy behind it. In particular, we feel strongly that APRS is, and should remain, a light-weight tactical system — almost anyone should be able to use it in temporary situations (such as emergencies or mobile work or weather watching) with the minimum of training and equipment.

This document is the result of inputs from many people, and collated and massaged by the APRS Working Group. Our sincere thanks go to everyone who has contributed in putting it together and getting it onto the street. If you discover any errors or omissions or misleading statements, please let us know — the best way to do this is via the TAPR *aprsspec* mailing list at www.tapr.org.

Finally, users throughout the world are continually coming up with new ideas and suggestions for extending and improving APRS. We welcome them. Again, the best way to discuss these is via the *aprsspec* list.

The APRS Working Group

August 2000

Disclaimer

Like any navigation system, APRS is not infallible. No one should rely blindly on APRS for navigation, or in life-and-death situations. Similarly, this specification is not infallible.

The members of the APRS Working Group have done their best to define the APRS protocol, but this protocol description may contain errors, or there may be omissions. It is very likely that not all APRS implementations will fully or correctly implement this specification, either today or in the future.

We urge anyone using or writing a program that implements this specification to exercise caution and good judgement. The APRS Working Group and the specification's Editor disclaim all liability for injury to persons or property that may result from the use of this specification or software implementing it.

THE STRUCTURE OF THIS SPECIFICATION

This specification describes the overall requirements for developing software that complies with APRS Protocol Version 1.0. The information flow starts with the standard AX.25 UI-frame, and progresses downwards into more and more detail as the use of each field in the frame is explored.

A key feature of the specification is the inclusion of dozens of detailed examples of typical APRS packets and related math computations.

Here is an outline of the chapters:

Introduction to APRS — A brief background to APRS and a summary of its main features.

The APRS Design Philosophy — The fundamentals of APRS, highlighting its use as a real-time tactical communications tool, the timing of APRS transmissions and the use of generic digipeating.

APRS and AX.25 — A brief refresher on the structure of the AX.25 UI-frame, with particular reference to the special ways in which APRS uses the Destination and Source Address fields and the Information field.

APRS Data in the AX.25 Destination and Source Address Fields — Details of generic APRS callsigns and callsigns that specify display symbols and APRS software version numbers. Also a summary of how Mic-E encoded data is stored in the Destination Address field, and how the Source Address SSID can specify a display icon.

APRS Data in the AX.25 Information Field — Details of the principal constituents of APRS data that are stored in the Information field. Contains the APRS Data Type Identifiers table, and a summary of all the different types of data that the Information field can hold.

Time and Position Formats — Information on formats for timestamps, latitude, longitude, position ambiguity, Maidenhead locators, NMEA data and altitude.

APRS Data Extensions — Details of optional data extensions for station course/speed, wind speed/direction, power/height/gain, pre-calculated radio range, DF signal strength and Area Object descriptor.

Position and DF Report Data Formats — Full details of these report formats.

Compressed Position Report Data Formats — Full details of how station position and APRS data extensions are compressed into very short packets.

Mic-E Data Format —Mic-E encoding of station lat/long position, altitude, course, speed, Mic-E message code, telemetry data and APRS digipeater path into the AX.25 Destination Address and Information fields.

Object and Item Reports — Full information on how to set up APRS Objects and Items, and details of the encoding of Area Objects (circles, lines, ellipses etc).

Weather Reports — Full format details for weather reports from stand-alone (positionless) weather stations and for reports containing position information. Also details of storm data format.

Telemetry Data — A description of the MIM/KPC-3+ telemetry data format, with supporting information on how to tailor the interpretation of the raw data to individual circumstances.

Messages, Bulletins and Announcements — Full format information.

Station Capabilities, Queries and Responses — Details of the ten different types of query and expected responses.

Status Reports — The format of general status messages, plus the special cases of using a status report to contain meteor scatter beam heading/power and Maidenhead locator.

Network Tunneling — The use of the Source Path Header to allow tunneling of APRS packets through third-party networks that do not understand AX.25 addresses, and the use of the third-party Data Type Identifier.

User-Defined Data Format — APRS allows users to define their own data formats for special purposes. This chapter describes how to do this.

Other Packets — A general statement on how APRS is to handle any other packet types that are not covered by this specification.

APRS Symbols —How to specify APRS symbols and symbol overlays, in position reports and in generic GPS destination callsigns.

APRS Data Formats — An appendix containing all the APRS data formats collected together for easy reference.

The APRS Symbol Tables —A complete listing of all the symbols in the Primary and Alternate Symbol Tables.

ASCII Code Table — The full ASCII code, including decimal and hex codes for each character (the decimal code is needed for compressed lat/long and altitude computations), together with the hex codes for bit-shifted ASCII characters in AX.25 addresses (useful for Mic-E decoding and general on-air packet monitoring).

Glossary — A handy one-stop reference for the many APRS-specific terms used in this specification.

References — Pointers to other documents that are relevant to this specification.

1 INTRODUCTION TO APRS

What is APRS?

APRS is short for *Automatic Position Reporting System*, which was designed by Bob Bruninga, WB4APR, and introduced by him at the 1992 TAPR/ ARRL Digital Communications Conference.

Fundamentally, APRS is a packet communications protocol for disseminating live data to everyone on a network in real time. Its most visual feature is the combination of packet radio with the Global Positioning System (GPS) satellite network, enabling radio amateurs to automatically display the positions of radio stations and other objects on maps on a PC. Other features not directly related to position reporting are supported, such as weather station reporting, direction finding and messaging.

APRS is different from regular packet in several ways:

- It provides maps and other data displays, for vehicle/personnel location and weather reporting in real time.
- It performs all communications using a one-to-many protocol, so that everyone is updated immediately.
- It uses generic digipeating, with well-known callsign aliases, so that prior knowledge of network topology is not required.
- It supports intelligent digipeating, with callsign substitution to reduce network flooding.
- Using AX.25 UI-frames, it supports two-way messaging and distribution of bulletins and announcements, leading to fast dissemination of text information.
- It supports communications with the Kenwood TH-D7 and TM-D700 radios, which have built-in TNC and APRS firmware.

Conventional packet radio is really only useful for passing bulk message traffic from point to point, and has traditionally been difficult to apply to real-time events where information has a very short lifetime. APRS turns packet radio into a real-time tactical communications and display system for emergencies and public service applications.

APRS provides universal connectivity to all stations, but avoids the complexity, time delays and limitations of a connected network. It permits any number of stations to exchange data just like voice users would on a voice net. Any station that has information to contribute simply sends it, and all stations receive it and log it.

APRS recognizes that one of the greatest real-time needs at any special event or emergency is the tracking of key assets. Where is the marathon leader? Where are the emergency vehicles? What's the weather at various points in the county? Where are the power lines down? Where is the head of the

parade? Where is the mobile ATV camera? Where is the storm?

To address these questions, APRS provides a fully featured automatic vehicle location and status reporting system. It can be used over any two-way radio system including amateur radio, marine band, and cellular phone. There is even an international live APRS tracking network on the Internet.

APRS Features

APRS runs on most platforms, including DOS, Windows 3.x, Windows 95/98, MacOS, Linux and Palm. Most implementations on these platforms support the main features of APRS:

- **Maps** — APRS station positions can be plotted in real-time on maps, with coverage from a few hundred yards to worldwide. Stations reporting a course and speed are dead-reckoned to their present position. Overlay databases of the locations of APRS digipeaters, US National Weather Service sites and even amateur radio stores are available. It is possible to zoom in to any point on the globe.
- **Weather Station Reporting** — APRS supports the automatic display of remote weather station information on the screen.
- **DX Cluster Reporting** — APRS an ideal tool for the DX cluster user. Small numbers of APRS stations connected to DX clusters can relay DX station information to many other stations in the local area, reducing overall packet load on the clusters.
- **Internet Access** — The Internet can be used transparently to cross-link local radio nets anywhere on the globe. It is possible to telnet into Internet APRS servers and see hundreds of stations from all over the world live. Everyone connected can feed their locally heard packets into the APRS server system and everyone everywhere can see them.
- **Messages** — Messages are two-way messages with acknowledgement. All incoming messages alert the user on arrival and are held on the message screen until killed.
- **Bulletins and Announcements** —Bulletins and announcements are addressed to everyone. Bulletins are sent a few times an hour for a few hours, and announcements less frequently but possibly over a few days.
- **Fixed Station Tracking** — In addition to automatically tracking mobile GPS/LORAN-equipped stations, APRS also tracks from manual reports or grid squares.
- **Objects** — Any user can place an APRS Object on his own map, and within seconds that object appears on all other station displays. This is particularly useful for tracking assets or people that are not equipped with trackers. Only one packet operator needs to know where things are (e.g. by monitoring voice traffic), and as he maintains the positions and movements of assets on his screen, all other stations running APRS will display the same information.

2 THE APRS DESIGN PHILOSOPHY

Net Cycle Time

It is important to note that APRS is primarily a *real-time, tactical* communications tool, to help the flow of information for things like special events, emergencies, Skywarn, the Emergency Operations Center and just plain in-the-field use under stress. But like the real world, for 99% of the time it is operating routinely, waiting for the unlikely serious event to happen.

Anything which is done to enhance APRS must not undermine its ability to operate in local areas under stress. Here are the details of that philosophy:

1. APRS uses the concept of a "net cycle time". This is the time within which a user should be able to hear (at least once) all APRS stations within range, to obtain a more or less complete picture of APRS activity. The net cycle time will vary according to local conditions and with the number of digipeaters through which APRS data travels.
2. The objective is to have a net cycle time of 10 minutes for local use. This means that within 10 minutes of arrival on the scene, it is possible to captured the entire tactical picture.
3. All stations, even fixed stations, should beacon their position at the net cycle time rate. In a stress situation, stations are coming and going all the time. The position reports show not only where stations are without asking, but also that they are still active.
4. It is not reasonable to assume that all APRS users responding to a stress event understand the ramifications of APRS and the statistics of the channel — user settings cannot be relied on to avoid killing a stressed net. Thus, to try to anticipate when the channel is under stress, APRS automatically adjusts its net cycle time according to the number of digipeaters in the UNPROTO path:
    - Direct operation (no digipeaters): 10 minutes (probably an event).
    - Via one digipeater hop: 10 minutes (probably an event).
    - Via two digipeater hops: 20 minutes.
    - Via three or more digipeater hops: 30 minutes.
5. Since almost all home stations set their paths to three or more digipeaters, the default net cycle time for routine daily operation is 30 minutes. This should be a universal standard that everyone can bank on — if you routinely turn on your radio and APRS and do nothing else, then in 30 minutes you should have virtually the total picture of all APRS stations within range.
6. Since knowing where the digipeaters are located is fundamental to APRS

connectivity, digipeaters should use multiple beacon commands to transmit position reports at different rates over different paths; i.e. every 10 minutes for sending position reports locally, and every 30 minutes for sending them via three digipeaters (plus others rates and distances as needed).

7. If the net cycle time is too long, users will be tempted to send queries for APRS stations. This will increase the traffic on the channel unnecessarily. Thus the recommended extremes for net cycle time are 10 and 30 minutes — this gives network designers the fundamental assumptions for channel loading necessary for good engineering design.

Packet Timing

Since APRS packets are error-free, but are not guaranteed delivery, APRS transmits information redundantly. To assure rapid delivery of new or changing data, and to preserve channel capacity by reducing interference from old data, APRS should transmit new information more frequently than old information.

There are several algorithms in use to achieve this:

- **Decay Algorithm** — Transmit a new packet once and n seconds later. Double the value of n for each new transmission. When n reaches the net cycle time, continue at that rate. Other factors besides "doubling" may be appropriate, such as for new message lines.
- **Fixed Rate** — Transmit a new packet once and n seconds later. Transmit it x times and stop.
- **Message-on-Heard** — Transmit a *new* packet according to either algorithm above. If the packet is still valid, and has not been acknowledged, and the net cycle time has been reached, then the recipient is probably not available. However, if a packet is then subsequently heard from the recipient, try once again to transmit the packet.
- **Time-Out** — This term is used to describe a time period beyond which it is reasonable to assume that a station no longer exists or is off the air if no packets have been heard from it. A period of 2 hours is suggested as the nominal default timeout. This time-out is not used in any transmitting algorithms, but is useful in some programs to decide when to cease displaying stations as "active". Note that on HF, signals come and go, so decisions about activity may need to be more flexible.

Generic Digipeating

The power of APRS in the field derives from the use of *generic* digipeating, in that packets are propagated without a priori knowledge of the network. There are six powerful techniques which have evolved since APRS was introduced in 1992:

1. **RELAY** — Every VHF APRS TNC is assumed to have an alias of `RELAY`, so that anyone can use it as a digipeater at any time.
2. **ECHO** — HF stations use the alias of `ECHO` as an alternative to RELAY. (However, bearing in mind the nature of HF propagation, this has the potential of causing interference over a wide area, and should only be used sparingly by mobile stations).
3. **WIDE** — Every high-site digipeater is assumed to have an alias of `WIDE` for longer distance communications.
4. **TRACE** — Every high-site digipeater that is using callsign substitution is assumed to have the alias of `TRACE`. These digipeaters self-identify packets they digipeat by inserting their own call in place of `RELAY`, `WIDE` or `TRACE`.
5. **WIDEn-N** — A digipeater that supports `WIDEn-N` digipeating will digipeat any `WIDEn-N` packet that is "new" and will subtract 1 from the SSID until the SSID reaches –0. The digipeater keeps a copy or a checksum of the packet and will not digipeat that packet again within (typically) 28 seconds. This considerably reduces the number of superfluous digipeats in areas with many digipeaters in radio range of each other.
6. **GATE** — This generic callsign is used by HF-to-VHF Gateway digipeaters. Any packet heard on HF via `GATE` will be digipeated locally on VHF. This permits local networks to keep an eye on the national and international picture.

Communicating Map Views Unambiguously

APRS is a tactical geographical system. To maximize its operational effectiveness and minimize confusion between operators of different systems, users need to have an unambiguous way to communicate to others the "location" and "size" (or area of coverage) of any map view.

The APRS convention is by reference to a *center* and *range* which specify the geographical center and approximate radius of a circle that will fit in the map view independent of aspect ratio. The radius of the circle (in nautical miles, statute miles or km) is known as the "range scale". This convention gives all users a simple common basis for describing any specific map view to others over any communications medium or program.

3 APRS AND AX.25

Protocols

At the link level, APRS uses the AX.25 protocol, as defined in *Amateur Packet-Radio Link-Layer Protocol* (see Appendix 6 for details), utilizing Unnumbered Information (UI) frames exclusively. This means that APRS runs in *connectionless* mode, whereby AX.25 frames are transmitted without expecting any response, and reception at the other end is not guaranteed.

At a higher level, APRS supports a messaging protocol that allows users to send short messages (one line of text) to nominated stations, and expects to receive acknowledgements from those stations.

The AX.25 Frame

All APRS transmissions use AX.25 UI-frames, with 9 fields of data:

	AX.25 UI-FRAME FORMAT								
	Flag	***Destination Address***	***Source Address***	***Digipeater Addresses (0-8)***	***Control Field (UI)***	***Protocol ID***	***INFORMATION FIELD***	***FCS***	***Flag***
Bytes:	1	7	7	0–56	1	1	1–256	2	1

- **Flag** — The flag field at each end of the frame is the bit sequence 0x7e that separates each frame.
- **Destination Address** — This field can contain an APRS destination callsign or APRS data. APRS data is encoded to ensure that the field conforms to the standard AX.25 callsign format (i.e. 6 alphanumeric characters plus SSID). If the SSID is non-zero, it specifies a generic APRS digipeater path.
- **Source Address** — This field contains the callsign and SSID of the transmitting station. In some cases, if the SSID is non-zero, the SSID may specify an APRS display Symbol Code.
- **Digipeater Addresses** — From zero to 8 digipeater callsigns may be included in this field. **Note**: These digipeater addresses may be overridden by a generic APRS digipeater path (specified in the Destination Address SSID).
- **Control Field** — This field is set to 0x03 (UI-frame).
- **Protocol ID** — This field is set to 0xf0 (no layer 3 protocol).
- **Information Field** — This field contains more APRS data. The first character of this field is the APRS Data Type Identifier that specifies the nature of the data that follows.
- **Frame Check Sequence** — The FCS is a sequence of 16 bits used for checking the integrity of a received frame.

4 APRS DATA IN THE AX.25 DESTINATION AND SOURCE ADDRESS FIELDS

The AX.25 Destination Address Field

The AX.25 Destination Address field can contain 6 different types of APRS information:

- A generic APRS address.
- A generic APRS address with a symbol.
- An APRS software version number.
- Mic-E encoded data.
- A Maidenhead Grid Locator (obsolete).
- An Alternate Net (ALTNET) address.

In all of these cases, the Destination Address SSID may specify a generic APRS digipeater path.

Generic APRS Destination Addresses

APRS uses the following generic beacon-style destination addresses:

AIR* †	**ALL***	**AP***	**BEACON**	**CQ***	**GPS***	**DF***
DGPS*	**DRILL***	**DX***	**ID***	**JAVA***	**MAIL***	**MICE***
QST*	**QTH***	**RTCM***	**SKY***	**SPACE***	**SPC***	**SYM***
TEL*	**TEST***	**TLM***	**WX***	**ZIP*** †		

The asterisk is a wildcard, allowing the address to be extended (up to a total of 6 alphanumeric characters). Thus, for example, `WX1`, `WX12` and `WX12CD` are all valid APRS destination addresses.

† The **AIR*** and **ZIP*** addresses are being phased out, but are needed at present for backward compatibility.

All of these addresses have an SSID of –0. Non-zero SSIDs are reserved for generic APRS digipeating.

These addresses are copied by everyone. All APRS software must accept packets with these destination addresses.

The address **GPS** (i.e. the 3-letter address **GPS**, not **GPS***) is specifically intended for use by trackers sending lat/long positions via digipeaters which have the capability of converting positions to compressed data format.

The addresses **DGPS** and **RTCM** are used by differential GPS correction stations. Most software will not make use of packets using this address, other than to pass them on to an attached GPS unit.

The address **SKY** is used for Skywarn stations.

Packets addressed to **SPCL** are intended for special events. APRS software can display such packets to the exclusion of all others, to minimize clutter on

the screen from other stations not involved in the special event.

The addresses **TEL** and **TLM** is used for telemetry stations.

Generic APRS Address with Symbol

APRS uses several of the above-listed generic addresses in a special way, to specify not only an address but also a display symbol. These special addresses are GPSxyz, GPSCnn, GPSEnn, SPCxyz and SYMxyz, and are intended for use where it is not possible to include the symbol in the AX.25 Information field.

The GPS addresses above are for general use.

The SPC addresses are intended for special events.

The SYM addresses are reserved for future use.

The characters xy and nn refer to entries in the APRS Symbol Tables. The character z specifies a symbol overlay. See Chapter 20: APRS Symbols and Appendix 2 for more information.

APRS Software Version Number

The AX.25 Destination Address field can contain the version number of the APRS software that is running at the station. Knowledge of the version number can be useful when debugging.

The following software version types are reserved (xx and xxx indicate a version number):

APCxxx	APRS/CE, Windows CE
APDxxx	Linux aprsd server
APExxx	PIC-Encoder
APIxxx	Icom radios (future)
APICxx	ICQ messaging
APKxxx	Kenwood radios
APMxxx	MacAPRS
APPxxx	pocketAPRS
APRxxx	APRSdos
APRS	older versions of APRSdos
APRSM	older versions of MacAPRS
APRSW	older versions of WinAPRS
APSxxx	APRS+SA
APWxxx	WinAPRS
APXxxx	X-APRS
APYxxx	Yaesu radios (future)
APZxxx	Experimental

This table will be added to by the APRS Working Group.

For example, a station using version 3.2.6 of MacAPRS could use the

destination callsign `APM326`.

The Experimental destination is designated for *temporary* use only while a product is being developed, before a special APRS Software Version address is assigned to it.

Mic-E Encoded Data

Another alternative use of the AX.25 Destination Address field is to contain Mic-E encoded data. This data includes:

- The latitude of the station.
- A West/East Indicator and a Longitude Offset Indicator (used in longitude computations).
- A Message Code.
- The APRS digipeater path.

This data is used with associated data in the AX.25 Information field to provide a complete Position Report and other information about the station (see Chapter 10: Mic-E Data Format).

Maidenhead Grid Locator in Destination Address

The AX.25 Destination Address field may contain a 6-character Maidenhead Grid Locator. For example: **`IO91SX`**. This format is typically used by meteor scatter and satellite operators who need to keep packets as short as possible.

This format is now obsolete.

Alternate Nets

Any other destination address not included in the specific generic list or the other categories mentioned above may be used in Alternate Nets (ALTNETs) by groups of individuals for special purposes. Thus they can use the APRS infrastructure for a variety of experiments without cluttering up the maps and lists of other APRS stations. Only stations using the same ALTNET address should see their data.

Generic APRS Digipeater Path

The SSID in the Destination Address field of all packets is coded to specify the APRS digipeater path.

If the Destination Address SSID is –0, the packet follows the standard AX.25 digipeater ("VIA") path contained in the Digipeater Addresses field of the AX.25 frame.

If the Destination Address SSID is non-zero, the packet follows one of 15 generic APRS digipeater paths.

The SSID field in the Destination Address (i.e. in the 7th address byte) is encoded as follows:

APRS Digipeater Paths in Destination Address SSID

SSID	*Path*	*SSID*	*Path*
-0	Use VIA path	**-8**	North path
-1	WIDE1-1	**-9**	South path
-2	WIDE2-2	**-10**	East path
-3	WIDE3-3	**-11**	West path
-4	WIDE4-4	**-12**	North path + WIDE
-5	WIDE5-5	**-13**	South path + WIDE
-6	WIDE6-6	**-14**	East path + WIDE
-7	WIDE7-7	**-15**	West path + WIDE

The AX.25 Source Address SSID to specify Symbols

The AX.25 Source Address field contains the callsign and SSID of the originating station. If the SSID is –0, APRS does not treat it in any special way.

If, however, the Source Address SSID is non-zero, APRS interprets it as a display icon. This is intended for use only with stand-alone trackers where there is no other method of specifying a display symbol or a destination address (e.g. MIM trackers or NMEA trackers).

For more information, see Chapter 20: APRS Symbols.

5 APRS DATA IN THE AX.25 INFORMATION FIELD

Generic Data Format

In general, the AX.25 Information field can contain some or all of the following information:

- APRS Data Type Identifier
- APRS Data
- APRS Data Extension
- Comment

	Generic APRS Information Field			
	Data Type ID	***APRS Data***	***APRS Data Extension***	***Comment***
Bytes:	1	n	7	n

APRS Data Type Identifier

Every APRS packet contains an APRS Data Type Identifier (DTI). This determines the format of the remainder of the data in the Information field, as follows:

APRS Data Type Identifiers

Ident	*Data Type*
0x1c	Current Mic-E Data (Rev 0 beta)
0x1d	Old Mic-E Data (Rev 0 beta)
!	Position without timestamp (no APRS messaging), or Ultimeter 2000 WX Station
"	*[Unused]*
#	Peet Bros U-II Weather Station
$	Raw GPS data or Ultimeter 2000
%	Agrelo DFJr / MicroFinder
&	*[Reserved — Map Feature]*
'	Old Mic-E Data (but *Current* data for TM-D700)
(	*[Unused]*
)	Item
*****	Peet Bros U-II Weather Station
+	*[Reserved — Shelter data with time]*
,	Invalid data or test data
-	*[Unused]*
.	*[Reserved — Space weather]*
/	Position with timestamp (no APRS messaging)
0–9	*[Do not use]*
:	Message
;	Object

Ident	*Data Type*
<	Station Capabilities
=	Position without timestamp (with APRS messaging)
>	Status
?	Query
@	Position with timestamp (with APRS messaging)
A–S	*[Do not use]*
T	Telemetry data
U–Z	*[Do not use]*
[	Maidenhead grid locator beacon (obsolete)
****	*[Unused]*
]	*[Unused]*
^	*[Unused]*
_	Weather Report (without position)
`	Current Mic-E Data (*not used* in TM-D700)
a–z	*[Do not use]*
{	User-Defined APRS packet format
|	*[Do not use — TNC stream switch character]*
}	Third-party traffic
~	*[Do not use — TNC stream switch character]*

Note: There is one exception to the requirement for the Data Type Identifier to be the *first* character in the Information field — this is the *Position without Timestamp* (indicated by the **!** DTI). The **!** character may occur *anywhere up to and including the 40th character position* in the Information field. This variability is required to support X1J TNC digipeaters which have a string of unmodifiable text at the beginning of the field.

Note: The Kenwood TM-D700 radio uses the **'** DTI for *current* Mic-E data. The radio does not use the **`** DTI.

APRS Data and Data Extension

There are 10 main types of APRS Data:

- Position
- Direction Finding
- Objects and Items
- Weather
- Telemetry
- Messages, Bulletins and Announcements
- Queries
- Responses
- Status
- Other

Some of this data may also have an APRS Data Extension that provides additional information.

The APRS Data and optional Data Extension follow the Data Type Identifier.

The table on the next page shows a complete list of all the different possible types of APRS Data and APRS Data Extension.

	Possible APRS Data	*Possible APRS Data Extension*
Position	Time (DHM or HMS) Lat/long coordinates Compressed lat/long/course/speed/radio range/altitude Symbol Table ID and Symbol Code Mic-E longitude, speed and course, telemetry or status Raw GPS NMEA sentence Raw weather station data	Course and Speed Power, Effective Antenna Height/Gain/Directivity Pre-Calculated Radio Range Omni DF Signal Strength Storm Data (in Comment field)
Direction Finding	Time (DHM or HMS) Lat/long coordinates Compressed lat/long/course/speed/radio range/altitude Symbol Table ID and Symbol Code	Course and Speed Power, Effective Antenna Height/Gain/Directivity Pre-Calculated Radio Range Omni DF Signal Strength Bearing and Number/Range/Quality (in Comment field)
Objects and Items	Object name Item name Time (DHM or HMS) Lat/long coordinates Compressed lat/long/course/speed/radio range/altitude Symbol Table ID and Symbol Code Raw weather station data	Course and Speed Power, Effective Antenna Height/Gain/Directivity Pre-Calculated Radio Range Omni DF Signal Strength Area Object Storm Data (in Comment field)
Weather	Time (MDHM) Lat/long coordinates Compressed lat/long/course/speed/radio range/altitude Symbol Table ID and Symbol Code Raw weather station data	Wind Direction and Speed Storm Data (in Comment field)
Telemetry	Telemetry (non Mic-E)	
Messages, Bulletins and Announcements	Addressee Message Text Message Identifier Message Acknowledgement Bulletin ID, Announcement ID Group Bulletin ID	
Queries	Query Type Query Target Footprint Addressee (Directed Query)	
Responses	Position Object/Item Weather Status Message Digipeater Trace Stations Heard Heard Statistics Station Capabilities	Course and Speed Power, Effective Antenna Height/Gain/Directivity Pre-Calculated Radio Range Omni DF Signal Strength Area Object Wind Direction and Speed
Status	Time (DHM zulu) Status text Meteor Scatter Beam Heading/Power Maidenhead Locator (Grid Square) Altitude (Mic-E) E-mail message	
Other	Third-Party forwarding Invalid Data/Test Data	

Comment Field

In general, any APRS packet can contain a plain text comment (such as a beacon message) in the Information field, immediately following the APRS Data or APRS Data Extension.

There is no separator between the APRS data and the comment unless otherwise stated.

The comment may contain any printable ASCII characters (except **|** and **~**, which are reserved for TNC channel switching).

The maximum length of the comment field depends on the report — details are included in the description of each report.

In special cases, the Comment field can also contain further APRS data:

- **Altitude** in comment text (see Chapter 6: Time and Position Formats), or in Mic-E status text (see Chapter 10: Mic-E Data Format).
- **Maidenhead Locator** (grid square), in a Mic-E status text field (see Chapter 10: Mic-E Data Format) or in a Status Report (see Chapter 16: Status Reports).
- **Bearing and Number/Range/Quality** parameters (`/BRG/NRQ`), in DF reports (see Chapter 7: APRS Data Extensions).
- **Area Object Line Widths** (see Chapter 11: Object and Item Reports).
- **Signpost Objects** (see Chapter 11: Object and Item Reports).
- **Weather and Storm Data** (see Chapter 12: Weather Reports).
- **Beam Heading and Power**, in Status Reports (see Chapter 16: Status Reports).

Base-91 Notation

Two APRS data formats use base-91 notation: lat/long coordinates in compressed format (see Chapter 9) and the altitude in Mic-E format (see Chapter 10).

Base-91 data is compressed into a short string of characters. All the characters are printable ASCII, with character codes in the range 33–124 decimal (i.e. **!** through **|**).

To compute the base-91 ASCII character string for a given data value, the value is divided by progressively reducing powers of 91 until the remainder is less than 91. At each step, 33 is added to the modulus of the division process to obtain the corresponding ASCII character code.

For example, for a data value of 12345678:

$12345678 / 91^3$ = modulus **16**, remainder 288542
$288542 / 91^2$ = modulus **34**, remainder 6988
$6988 / 91^1$ = modulus **76**, remainder **72**

The four ASCII character codes are thus 49 (i.e. **16**+33), 67 (i.e. **34**+33), 109 (i.e. **76**+33) and 105 (i.e. **72**+33), corresponding to the ASCII string **1Cmi**.

APRS Data Units

For historical reasons there is some lack of consistency between units of data in APRS packets — some speeds are in knots, others in miles per hour; some altitudes are in feet, others in meters, and so on. It is emphasized that this specification describes the units of data as they are transmitted on-air. It is the responsibility of APRS applications to convert the on-air units to more suitable units if required.

The default GPS earth datum is World Geodetic System (WGS) 1984.

6 TIME AND POSITION FORMATS

Time Formats

APRS timestamps are expressed in three different ways:

- Day/Hours/Minutes format
- Hours/Minutes/Seconds format
- Month/Day/Hours/Minutes format

In all three formats, the 24-hour clock is used.

Day/Hours/Minutes (DHM) format is a fixed 7-character field, consisting of a 6-digit *day/time* group followed by a single *time indicator* character (**z** or **/**). The day/time group consists of a two-digit day-of-the-month (01–31) and a four-digit time in hours and minutes.

Times can be expressed in *zulu* (UTC/GMT) or *local* time. For example:

`092345`**z** is 2345 hours *zulu* time on the 9th day of the month.
`092345`**/** is 2345 hours *local* time on the 9th day of the month.

It is recommended that future APRS implementations only transmit zulu format on the air.

Note: The time in Status Reports may *only* be in zulu format.

Hours/Minutes/Seconds (HMS) format is a fixed 7-character field, consisting of a 6-digit time in hours, minutes and seconds, followed by the **h** time-indicator character. For example:

`234517`**h** is 23 hours 45 minutes and 17 seconds *zulu*.

Note: This format may *not* be used in Status Reports.

Month/Day/Hours/Minutes (MDHM) format is a fixed 8-character field, consisting of the month (01–12) and day-of-the-month (01–31), followed by the time in hours and minutes zulu. For example:

`10092345` is 23 hours 45 minutes zulu on October 9th.

This format is only used in reports from stand-alone “positionless” weather stations (i.e. reports that do not contain station position information).

Use of Timestamps

When a station transmits a report *without* a timestamp, an APRS receiving station can make an internal record of the time it was received, if required. This record is the *receiving* station's notion of the time the report was created.

On the other hand, when a station transmits a report *with* a timestamp, that timestamp represents the *transmitting* station's notion of the time the report was created.

In other words, reports sent *without* a timestamp can be regarded as real-time, "current" reports (and the *receiving* station has to record the time they were received), whereas reports sent *with* a timestamp may or may not be real-time, and may possibly be (very) "old".

Four APRS Data Type Identifiers specify whether or not a report contains a timestamp, depending on whether the station has APRS messaging capability or not:

	No APRS Messaging	*With APRS Messaging*
(Current/real-time) Report without timestamp	**!**	**=**
(Old/non-real-time) Report with timestamp	**/**	**@**

Stations without APRS messaging capability are typically stand-alone trackers or digipeaters. Stations reporting without a timestamp are generally (but not necessarily) fixed stations.

Latitude Format

Latitude is expressed as a fixed 8-character field, in degrees and decimal minutes (to two decimal places), followed by the letter **N** for north or **S** for south.

Latitude degrees are in the range 00 to 90. Latitude minutes are expressed as whole minutes and hundredths of a minute, separated by a decimal point.

For example:

`4903.50`**N** is 49 degrees 3 minutes 30 seconds north.

In generic format examples, the latitude is shown as the 8-character string `ddmm.hhN` (i.e. degrees, minutes and hundredths of a minute north).

Longitude Format

Longitude is expressed as a fixed 9-character field, in degrees and decimal minutes (to two decimal places), followed by the letter **E** for east or **W** for west.

Longitude degrees are in the range 000 to 180. Longitude minutes are expressed as whole minutes and hundredths of a minute, separated by a decimal point.

For example:

`07201.75W` is 72 degrees 1 minute 45 seconds west.

In generic format examples, the longitude is shown as the 9-character string `dddmm.hhW` (i.e. degrees, minutes and hundredths of a minute west).

Position Coordinates

Position coordinates are a combination of latitude and longitude, separated by a display Symbol Table Identifier, and followed by a Symbol Code. For example:

`4903.50N/07201.75W-`

The `/` character between latitude and longitude is the Symbol Table Identifier (in this case indicating use of the Primary Symbol Table), and the `-` character at the end is the Symbol Code from that table (in this case, indicating a "house" icon).

A description of display symbols is included in Chapter 20: APRS Symbols. The full Symbol Table listing is in Appendix 2.

Position Ambiguity

In some instances — for example, where the exact position is not known — the sending station may wish to reduce the number of digits of precision in the latitude and longitude. In this case, the `mm` and `hh` digits in the latitude may be progressively replaced by a `␣` (space) character as the amount of imprecision increases. For example:

`4903.5␣N` represents latitude to nearest 1/10th of a minute.

`4903.␣␣N` represents latitude to nearest minute.

`490␣.␣␣N` represents latitude to nearest 10 minutes.

`49␣␣.␣␣N` represents latitude to nearest degree.

The level of ambiguity specified in the latitude will automatically apply to the longitude as well — it is not necessary to include any `␣` characters in the longitude.

For example, the coordinates:

`4903.␣␣N/07201.75W-`

represent the position to the nearest minute. That is, the hundredths of minutes of latitude and longitude may take any value in the range `00–99`.

Thus the station may be located anywhere inside a bounding box having the following corner coordinates:

North West corner: 49 deg 3.99 mins N, 72 deg 1.99 mins W
North East corner: 49 deg 3.99 mins N, 72 deg 1.00 mins W
South East corner: 49 deg 3.00 mins N, 72 deg 1.00 mins W
South West corner: 49 deg 3.00 mins N, 72 deg 1.99 mins W

Default Null Position

Where a station does not have *any* specific position information to transmit (for example, a Mic-E unit without a GPS receiver connected to it), the station must transmit a default null position in the location field.

The null position corresponds to 0° 0' 0" north, 0° 0' 0" west.

The null position should be include the **\.** symbol (unknown/indeterminate position). For example, a Position Report for a station with unknown position will contain the coordinates …`0000.00N`**`\`**`00000.00W`**`.`**…

Maidenhead Locator (Grid Square)

An alternative method of expressing a station's location is to provide a Maidenhead locator (grid square). There are four ways of doing this:

- In a Status Report — e.g. `IO91SX/-`
  (`/-` represents the symbol for a "house").
- In Mic-E Status Text — e.g. `IO91SX`**`/G`**
  (**`/G`** indicates a "grid square").
- In the Destination Address — e.g. `IO91SX`. (obsolete).
- In AX.25 beacon text, with the **`[`** APRS Data Type Identifier — e.g. **`[`**`IO91SX]` (obsolete).

Grid squares may be in 6-character form (as above) or in the shortened 4-character form (e.g. `IO91`).

NMEA Data

APRS recognizes raw ASCII data strings conforming to the NMEA 0183 Version 2.0 specification, originating from navigation equipment such as GPS and LORAN receivers. It is recommended that APRS stations interpret at least the following NMEA Received Sentence types:

GGA	Global Positioning System Fix Data
GLL	Geographic Position, Latitude/Longitude Data
RMC	Recommended Minimum Specific GPS/Transit Data
VTG	Velocity and Track Data
WPT	Way Point Location

Altitude

Altitude may be expressed in two ways:

- In the comment text.
- In Mic-E format.

Altitude in Comment Text — The comment may contain an altitude value, in the form **/A=**`aaaaaa`, where `aaaaaa` is the altitude in feet. For example: `/A=001234`. The altitude may appear anywhere in the comment.

Altitude in Mic-E format — The optional Mic-E status field can contain altitude data. See Chapter 10: Mic-E Data Format.

7 APRS DATA EXTENSIONS

A fixed-length 7-byte field may follow APRS position data. This field is an APRS Data Extension. The extension may be one of the following:

- `CSE/SPD` Course and Speed (this may be followed by a further 8 bytes containing DF bearing and Number/Range/Quality parameters)
- `DIR/SPD` Wind Direction and Wind Speed
- **`PHG`**`phgd` Station Power and Effective Antenna Height/Gain/ Directivity
- **`RNG`**`rrrr` Pre-Calculated Radio Range
- **`DFS`**`shgd` DF Signal Strength and Effective Antenna Height/Gain
- `Tyy/Cxx` Area Object Descriptor

Course and Speed

The 7-byte `CSE/SPD` Data Extension can be used to represent the course and speed of a vehicle or APRS Object.

The course is expressed in degrees (001-360), clockwise from due north. The speed is expressed in knots. A slash `/` character separates the two.

For example:

`088/036` represents a course 88 degrees, traveling at 36 knots.

If the course and speed are unknown or not relevant, they can be set to **`000/000`** or `.../...` or `___/___`.

Note: In the special case of DF reports, a course of `000` means that the DF station is fixed. If the course is non-zero, the station is mobile.

Wind Direction and Wind Speed

The 7-byte `DIR/SPD` Data Extension can be used to represent the wind direction and sustained one-minute wind speed in a Weather Report.

The wind direction is expressed in degrees (001-360), clockwise from due north. The speed is expressed in knots. A slash `/` character separates the two.

For example:

`220/004` represents a wind direction of 220 degrees and a speed of 4 knots.

If the wind direction and speed are unknown or not relevant, they can be set to **`000/000`** or `.../...` or `___/___`.

Power, Effective Antenna Height/Gain/ Directivity

The 7-byte **PHG**`phgd` Data Extension specifies the transmitter power, effective antenna height-above-average-terrain, antenna gain and antenna directivity. APRS uses this information to plot radio range circles around stations.

The 7 characters of this Data Extension are encoded as follows:

Characters 1–3: **PHG** (fixed)
Character 4: `p` Power code
Character 5: `h` Height code
Character 6: `g` Antenna gain code
Character 7: `d` Directivity code

The PHG codes are listed in the table below:

PHG Codes

`phgd` ***Code:***	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***Units***
Power	0	1	4	9	16	25	36	49	64	81	watts
Height	10	20	40	80	160	320	640	1280	2560	5120	feet
Gain	0	1	2	3	4	5	6	7	8	9	dB
Directivity	omni	45 NE	90 E	135 SE	180 S	225 SW	270 W	315 NW	360 N		deg

The height code represents the effective height of the antenna *above average local terrain*, not above ground or sea level — this is to provide a rough indication of the antenna's effectiveness in the local area .

The height code may in fact be any ASCII character 0–9 and above. This is so that larger heights for balloons, aircraft or satellites may be specified.

For example:
: is the height code for 10240 feet (approximately 1.9 miles).
; is the height code for 20480 feet (approximately 3.9 miles), and so on.

The Directivity code offsets the PHG circle by one third in the indicated direction. This means a front-to-back ratio of 2 to 1. Most often this is used to indicate a favored direction or a null, even if an omni antenna is at the site.

An example of the PHG Data Extension:

`PHG5132` means a power of 25 watts,
an antenna height of 20 feet above the average local terrain,
an antenna gain of 3 dB,
and maximum gain due east.

Range Circle Plot

On receipt, APRS uses the **p**, **h**, **g** and **d** codes to calculate the usable radio range (in miles), for plotting a range circle representing the local radio horizon around the station. The radio range is calculated as follows:

power = $\mathbf{p}^2$

Height-above-average-terrain (haat) = $10 \times 2^{\mathbf{h}}$

gain = $10^{(\mathbf{g}/10)}$

range = $\sqrt{(2 \times \text{haat} \times \sqrt{((\text{power}/10) \times (\text{gain}/2)) })}$

Thus, for `PHG5132`:

power = $\mathbf{5}^2 = 25$ watts

haat = $10 \times 2^{\mathbf{1}} = 20$ feet

gain = $10^{(\mathbf{3}/10)} = 1.995262$

range = $\sqrt{(2 \times 20 \times \sqrt{((25/10) \times (1.995262/2)) })}$

~ 7.9 miles

As the direction of maximum gain is due east, APRS will draw a range circle of radius 8 miles around the station, offset by 2.7 miles (i.e. one third of 8 miles) in an easterly direction.

Note: In the absence of any PHG data, stations are assumed to be running 10 watts to a 3dB omni antenna at 20 feet, resulting in a 6-mile radius range circle, centered on the station.

Pre-Calculated Radio Range

The 7-byte **RNG**`rrrr` Data Extension allows users to transmit a pre-calculated omni-directional radio range, where `rrrr` is the range in miles (with leading zeros).

For example, `RNG0050` indicates a radio range of 50 miles.

APRS can use this value to plot a range circle around the station.

Omni-DF Signal Strength

The 7-byte **DFS**`shgd` Data Extension lets APRS localize jammers by plotting the overlapping signal strength contours of all stations hearing the signal. This Omni-DF format replaces the PHG format to indicate DF signal strength, in that the transmitter power field is replaced with the relative signal strength (s) from 0 to 9.

DFS Codes

shgd ***Code:***	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***Units***
Strength	0	1	2	3	4	5	6	7	8	9	S-points
Height	10	20	40	80	160	320	640	1280	2560	5120	feet
Gain	0	1	2	3	4	5	6	7	8	9	dB
Directivity	omni	45 NE	90 E	135 SE	180 S	225 SW	270 W	315 NW	360 N		deg

For example, `DFS2360` represents a weak signal (around strength S2) heard on an omni antenna with 6 dB gain at 80 feet.

A signal strength of zero (`0`) is particularly significant, because APRS uses these `0` signal reports to draw (usually black) circles where the jammer is *not* heard. These black circles are extremely valuable since there will be a lot more reports from stations that do not hear the jammer than from those that do. This quickly eliminates a lot of territory.

Bearing and Number/Range/ Quality

DF reports contain an 8-byte field `/BRG/NRQ` that follows the `CSE/SPD` Data Extension, specifying the course, speed, bearing and NRQ (Number/Range/ Quality) value of the report. NRQ indicates the Number of hits, the approximate Range and the Quality of the report.

For example, in:

`…088/036/270/729…` course = 88 degrees, speed = 36 knots, bearing = 270 degrees, N = 7, R = 2, Q = 9

If N is 0, then the NRQ value is meaningless. Values of N from 1 to 8 give an indication of the number of hits per period relative to the length of the time period — thus a value of 8 means 100% of all samples possible got a hit. A value of 9 for N indicates to other users that the report is manual.

The N value is not processed, but is just another indicator from the automatic DF units.

The range limits the length of the line to the original map's scale of the sending station. The range is 2^R so, for R=4, the range will be 16 miles.

Q is a single digit in the range 0–9, and provides an indication of bearing accuracy:

Q	***Bearing Accuracy***
0	Useless
1	< 240 deg
2	< 120 deg
3	< 64 deg
4	< 32 deg

Q	***Bearing Accuracy***
5	< 16 deg
6	< 8 deg
7	< 4 deg
8	< 2 deg
9	< 1 deg (best)

If the course and speed parameters are not appropriate, they should have the value **000/000** or .../... or ___/___.

Area Object Descriptor

The 7-byte Tyy/Cxx Data Extension is an Area Object Descriptor. The T parameter specifies the type of object (square, circle, triangle, etc) and the /C parameter specifies its fill color.

Area Objects are described in Chapter 11: Object and Item Reports.

8 POSITION AND DF REPORT DATA FORMATS

Position Reports

Lat/Long Position Reports are contained in the Information field of an APRS AX.25 frame.

The following diagrams show the permissible formats of these reports, together with some examples. The gray areas indicate optional fields, and the shaded (yellow) characters are literal ASCII characters. In all cases there is a maximum of 43 characters after the Symbol Code.

Lat/Long Position Report Format — without Timestamp

	! or =	***Lat***	***Sym Table ID***	***Long***	***Symbol Code***	***Comment (max 43 chars)***
Bytes:	1	8	1	9	1	0-43

Examples

`!4903.50N/07201.75W-Test 001234`	no timestamp, no APRS messaging, with comment.
`!4903.50N/07201.75W-Test /A=001234`	no timestamp, no APRS messaging, altitude = 1234 ft.
`!49␣␣.␣␣N/072␣␣.␣␣W-`	no timestamp, no APRS messaging, location to nearest degree.
`TheNet␣X-1J4␣␣(BFLD)``!4903.50N/07201.75Wn`	no timestamp, no APRS messaging, with **X1J** node header string.

Lat/Long Position Report Format — with Timestamp

	/ or @	***Time DHM / HMS***	***Lat***	***Sym Table ID***	***Long***	***Symbol Code***	***Comment (max 43 chars)***
Bytes:	1	7	8	1	9	1	0-43

Examples

`/092345z4903.50N/07201.75W>Test1234`	with timestamp, no APRS messaging, zulu time, with comment.
`@092345/4903.50N/07201.75W>Test1234`	with timestamp, with APRS messaging, local time, with comment.

Lat/Long Position Report Format — with Data Extension (no Timestamp)							
	!** or **=	***Lat***	***Sym Table ID***	***Long***	***Symbol Code***	***Course/Speed*** ***Power/Height/Gain/Dir*** ***Radio Range*** ***DF Signal Strength***	***Comment (max 36 chars)***
Bytes:	1	8	1	9	1	7	0-36

Example

```
=4903.50N/07201.75W#PHG5132          no timestamp, with APRS messaging, with PHG.
=4903.50N/07201.75W_225/000g000t050r000p001...h00b10138dU2k   weather report.
```

Lat/Long Position Report Format — with Data Extension and Timestamp								
	/** or **@	***Time DHM / HMS***	***Lat***	***Sym Table ID***	***Long***	***Symbol Code***	***Course/Speed*** ***Power/Height/Gain/Dir*** ***Radio Range*** ***DF Signal Strength***	***Comment (max 36 chars)***
Bytes:	1	7	8	1	9	1	7	0-36

Examples

```
@092345/4903.50N/07201.75W>088/036   with timestamp, with APRS messaging, local time,
                                     course/speed.
@234517h4903.50N/07201.75W>PHG5132   with timestamp, APRS messaging, hours/mins/secs
                                     time, PHG.
@092345z4903.50N/07201.75W>RNG0050   with timestamp, APRS messaging, zulu time, radio
                                     range.
/234517h4903.50N/07201.75W>DFS2360   with timestamp, hours/mins/secs time, DF,
                                     no APRS messaging.
@092345z4903.50N/07201.75W_090/000g000t066r000p000...dUII   weather report.
```

Maidenhead Locator Beacon			
	[	***Grid Locator***	***]***	***Comment***
Bytes:	1	4 or 6	1	n

Examples

```
[IO91SX] 35 miles NNW of London
[IO91]
```

Raw NMEA Position Report Format	
NMEA Received Sentence	
$	…,…,…,…,…,…,…
Bytes: 1	25-209

Examples

```
$GPGGA,102705,5157.9762,N,00029.3256,W,1,04,2.0,75.7,M,47.6,M,,*62
$GPGLL,2554.459,N,08020.187,W,154027.281,A
$GPRMC,063909,A,3349.4302,N,11700.3721,W,43.022,89.3,291099,13.6,E*52
$GPVTG,318.7,T,,M,35.1,N,65.0,K*69
```

DF Reports

DF Reports are contained in the Information field of an APRS AX.25 frame. The Bearing and Number/Range/Quality (BRG/NRQ) parameters follow the Data Extension field.

Note: The BRG/NRQ parameters are only meaningful when the report contains the DF symbol (i.e. the Symbol Table ID is **/** and the Symbol Code is ****).

Note: If the DF station is fixed, the Course value is zero. If the station is moving, the Course value is non-zero.

DF Report Format — without Timestamp							
! *or* **=**	*Lat*	*Sym Table ID* **/**	*Long*	*Symbol Code* ****	*Course/Speed* *Power/Height/Gain/Dir* *Radio Range* *DF Signal Strength*	**/***BRG***/***NRQ*	*Comment (max 28 chars)*
Bytes: 1	8	1	9	1	7	8	0-28

Examples

`=4903.50N/07201.75W\088/036/270/729`	no timestamp, course/speed/ bearing/NRQ, with APRS messaging. DF station moving (CSE is non-zero).
`=4903.50N/07201.75W\`**`000`**`/036/270/729`	Same report, DF station fixed (CSE=**000**).

	DF Report Format — with Timestamp								
	/ *or* ***@***	***Time DHM / HMS***	***Lat***	***Sym Table ID /***	***Long***	***Symbol Code ***	***Course/Speed*** ***Power/Height/Gain/Dir*** ***Radio Range*** ***DF Signal Strength***	***/BRG/NRQ***	***Comment (max 28 chars)***
Bytes:	1	7	8	1	9	1	7	8	0-28

Examples

`@092345z4903.50N/07201.75W\088/036/270/729` with timestamp, course/speed/ bearing/NRQ, with APRS messaging.

`/092345z4903.50N/07201.75W\000/000/270/729` with timestamp, bearing/NRQ, no course/speed, no APRS messaging.

9 COMPRESSED POSITION REPORT DATA FORMATS

In compressed data format, the Information field contains the station's latitude and longitude, together with course and speed or pre-calculated radio range or altitude.

This information is compressed to minimize the length of the transmitted packet (and therefore improve its chances of being received correctly under less than ideal conditions).

The Information field also contains a display Symbol Code, and there may optionally be a plain text comment (uncompressed) as well.

The Advantages of Data Compression

Compressed data format may be used in place of the numeric lat/long coordinates already described, such as in the **!**, **/**, **@** and **=** formats.

Data compression has several important benefits:

- Fully backwards compatible with all existing formats.
- Fully supports any comment string.
- Speed is accurate to +/-1 mph up to about 40 mph and within 3% at 600 mph.
- Altitude in feet is accurate to +/- 0.4% from 1 foot to 3000 miles.
- Consistent one-algorithm processing of compressed latitude and longitude.
- Improved position to 1 foot worldwide.
- Pre-calculated radio range, compressed to one byte.
- Potential 50% compression of every position format on the air.
- Potential 40% reduction of raw GPS NMEA data length.
- Additional 7-byte reduction for NEMA GGA altitudes.
- Support for TNC compression at the NMEA source (from the GPS receiver).
- Digipeater compression of old NMEA trackers on the fly.
- Usage is optional in all cases.

The only minor disadvantages are that the course only resolves to +/- 2 degrees, and this format does not support PHG.

Compressed Data Format

Compressed data may be generated in several ways:

- by APRS software.
- pre-entered manually into a digipeater's beacon text.
- by a digipeater converting raw tracker NMEA packets to compressed.

[In future, there is the possibility that a Kantronics KPC-3 or other tracker TNC will be able to compress data directly from an attached GPS receiver].

In all cases the compressed format is a fixed 13-character field:

```
/YYYYXXXX$csT
```

where `/` is the Symbol Table Identifier
`YYYY` is the compressed latitude
`XXXX` is the compressed longitude
`$` is the Symbol Code
`cs` is the compressed course/speed or compressed pre-calculated radio range or compressed altitude
`T` is the compression type indicator

Compressed Position Data					
Sym Table ID	***Compressed Lat*** `YYYY`	***Compressed Long*** `XXXX`	***Symbol Code***	***Compressed Course/Speed*** / ***Compressed Radio Range*** / ***Compressed Altitude***	***Comp Type*** `T`
Bytes: 1	4	4	1	2	1

Compressed format can be used in place of lat/long position format anywhere that …`ddmm.hhN/dddmm.hhW$xxxxxxx`… occurs.

All bytes except for the **/** and **$** are base-91 printable ASCII characters (**!**..**{**). These are converted to numeric values by subtracting 33 from the decimal ASCII character code. For example, **#** has an ASCII code of 35, and represents a numeric value of 2 (i.e. 35-33).

Symbol

The presence of the leading Symbol Table Identifier instead of a digit indicates that this is a compressed Position Report and not a normal lat/long report.

Lat/Long Encoding

The values of YYYY and XXXX are computed as follows:

YYYY is 380926 x (90 – latitude) [base 91]
latitude is positive for north, negative for south, in degrees.

XXXX is 190463 x (180 + longitude) [base 91]
longitude is positive for east, negative for west, in degrees.

For example, for a longitude of 72° 45' 00" west (i.e. -72.75 degrees), the math is 190463 x (180 – 72.75) = 20427156. Because this is to base 91, it is then necessary to progressively divide this value by reducing powers of 91, to obtain the numerical values of X:

$20427156 / 91^3$ = **27**, remainder 80739
$80739 / 91^2$ = **9**, remainder 6210
$6210 / 91^1$ = **68**, remainder **22**

To obtain the corresponding ASCII characters, 33 is added to each of these values, yielding 60 (i.e. 27+33), 42, 101 and 55. From the ASCII Code Table (in Appendix 3), this corresponds to **<*e7** for XXXX.

Lat/Long Decoding

To decode a compressed lat/long, the reverse process is needed. That is, if YYYY is represented as $y_1y_2y_3y_4$ and XXXX as $x_1x_2x_3x_4$, then:

$$Lat = 90 - ((y_1-33) \times 91^3 + (y_2-33) \times 91^2 + (y_3-33) \times 91 + y_4-33) / 380926$$

$$Long = -180 + ((x_1-33) \times 91^3 + (x_2-33) \times 91^2 + (x_3-33) \times 91 + x_4-33) / 190463$$

For example, if the compressed value of the longitude is **<*e7** (as computed above), the calculation becomes:

$$\begin{aligned} Long &= -180 + (27 \times 91^3 + 9 \times 91^2 + 68 \times 91 + 22) / 190463 \\ &= -180 + (20346417 + 74529 + 6188 + 22) / 190463 \\ &= -180 + 107.25 \\ &= -72.75 \text{ degrees} \end{aligned}$$

Course/Speed, Pre-Calculated Radio Range and Altitude

The two cs bytes following the Symbol Code character can contain either the compressed course and speed or the compressed pre-calculated radio range or the station's altitude. These two bytes are in base 91 format.

In the special case of c = ␣ (space), there is no course, speed or range data, in which case the csT bytes are ignored.

Course/Speed — If the ASCII code for c is in the range **!** to **z** inclusive — corresponding to numeric values in the range 0–89 decimal (i.e. after subtracting 33 from the ASCII code) — then cs represents a compressed course/speed value:

course = **c** x 4

speed = $1.08^{\mathbf{s}} - 1$

For example, if the cs characters are **7P**, the corresponding values of **c** and **s** (after subtracting 33 from the ASCII character code) are 22 and 47 respectively. Substituting these values in the above equations:

course = **22** x 4 = 88 degrees

speed = $1.08^{\mathbf{47}} - 1$ = 36.2 knots

Pre-Calculated Radio Range — If c = **{**, then cs represents a compressed pre-calculated radio range value:

range = 2 x $1.08^{\mathbf{s}}$

For example, if the cs bytes are **{?**, the ASCII code for **?** is 63, so the value of **s** is 30 (i.e. 63-33). Thus:

range = 2 x $1.08^{\mathbf{30}}$

~ 20 miles

So APRS will draw a circle of radius 20 miles around the station plot on the screen.

The Compression Type (T) Byte

The T byte follows the cs bytes. The T byte contains several bit fields showing the GPS fix status, the NMEA source of the position data and the origin of the compression.

The T byte is not meaningful if the c byte is **_** (space).

	Compression Type (T) Byte Format							
Bit:	7	6	5	4	3	2	1	0
	Not used	*Not used*	***GPS Fix***	***NMEA Source***		***Compression Origin***		
Value:	0	0	0 = old (last) 1 = current	0 0 = other 0 1 = GLL 1 0 = GGA 1 1 = RMC		0 0 0 = Compressed 0 0 1 = TNC BText 0 1 0 = Software (DOS/Mac/Win/+SA) 0 1 1 = [tbd] 1 0 0 = KPC3 1 0 1 = Pico 1 1 0 = Other tracker [tbd] 1 1 1 = Digipeater conversion		

For example, if the compressed position was derived from an RMC sentence, the fix is current, and the compression was performed by APRSdos software, then the value of T in binary is 0 0 1 11 010, which equates to 58 decimal. Adding 33 to this value gives the ASCII code for the T byte (i.e. 91), which

corresponds to the **[** character.

Thus, using data from all the earlier examples, if the RMC sentence contains (among other parameters) the following data:

Latitude = 49° 30' 00" north
Longitude = 72° 45' 00" west
Speed = 36.2 knots
Course = 88°

and: the fix is current
compression is performed by APRSdos software
the display symbol is a "car"

then the complete 13-character compressed location field is transmitted as:

/	YYYY	XXXX	$	csT
/	**5L!!**	**<*e7**	**>**	**7P[**

Altitude

If the T byte indicates that the raw data originates from a GGA sentence (i.e. bits 4 and 3 of the T byte are 10), then the sentence contains an altitude value, in feet. After compression, the compressed altitude data is placed in the cs bytes, such that:

altitude = 1.002^{cs} feet

For example, if the received cs bytes are **S]**, the computation is as follows:

- Subtract 33 from the ASCII code for each character:
  c = 83 – 33 = 50
  s = 93 – 33 = 60
- Multiply **c** by 91 and add **s** to obtain **cs**:
  cs = 50 x 91 + 60
  = 4610
- Then altitude = 1.002^{4610}
  = 10004 feet

New Trackers

Tracker firmware may compress GPS data directly to APRS compressed format. They would use the **!** Data Type Indicator, showing that the position is real-time and that the tracker is not APRS-capable.

If the Position Report is not real-time, then the **/** Data Type Indicator can be used instead, so that the latest fix time may be included.

Old Trackers

Some digipeaters have the ability to convert raw NMEA strings from existing trackers to compressed data format for further forwarding.

These digipeaters will compress the data if the tracker Destination Address is `GPS`. (**Note**: This is the 3-letter address `GPS`, not `GPS*`).

Trackers desiring for their packets to not be modified by the APRS network will use any other valid generic APRS Destination Address.

Compressed Report Formats

Compressed data is contained in the AX.25 Information field, in these formats:

Compressed Lat/Long Position Report Format — no Timestamp

	!** or **=	***Sym Table ID***	***Comp Lat*** `YYYY`	***Comp Long*** `XXXX`	***Symbol Code***	***Compressed Course/Speed*** / ***Compressed Radio Range*** / ***Compressed Altitude***	***Comp Type*** `T`	***Comment (max 40 chars)***
Bytes:	1	1	4	4	1	2	1	0-40

Examples

`=/5L!!<*e7>␣sTComment`	with APRS messaging. Note the ␣ space character following the > Symbol Code, indicating that there is no course/speed, radio range or altitude. The **sT** characters are fillers and have no significance here.
`=/5L!!<*e7>7P[`	with APRS messaging, RMC sentence, with course/speed.
`=/5L!!<*e7>{?!`	with APRS messaging, with radio range.
`=/5L!!<*e7OS]S`	with APRS messaging, GGA sentence, altitude.

Compressed Lat/Long Position Report Format — with Timestamp

	/** or **@	***Time DHM / HMS***	***Sym Table ID***	***Comp Lat*** `YYYY`	***Comp Long*** `XXXX`	***Symbol Code***	***Compressed Course/Speed*** / ***Compressed Radio Range*** / ***Compressed Altitude***	***Comp Type*** `T`	***Comment (max 40 chars)***
Bytes:	1	7	1	4	4	1	2	1	0-40

Example

`@092345z/5L!!<*e7>{?!`	with APRS messaging, timestamp, radio range.

10 MIC-E DATA FORMAT

Mic-E Data Format

In Mic-E data format, the station's position, course, speed and display symbol, together with an APRS digipeater path and Mic-E Message Code, are packed into the AX.25 Destination Address and Information fields.

The Information field can also optionally contain either Mic-E telemetry data or Mic-E status. The Mic-E Status can contain the station's Maidenhead locator and altitude.

Mic-E packets can be very short. At the minimum, with no callsigns in the Digipeater Addresses field and no optional telemetry data or Mic-E status text, a complete Mic-E packet is just 25 bytes long (excluding FCS and flags).

Mic-E data format is not only used in the Microphone Encoder unit; it is also used in the PIC Encoder and in the Kenwood TH-D7 and TM-D700 radios.

Mic-E Data Payload

The Mic-E data format allows a large amount of data to be carried in a very short packet. The data is split between the Destination Address field and the Information field of a standard AX.25 UI-frame.

Destination Address Field — The 7-byte Destination Address field contains the following encoded information:

- The 6 latitude digits.
- A 3-bit Mic-E message identifier, specifying one of 7 Standard Mic-E Message Codes or one of 7 Custom Message Codes or an Emergency Message Code.
- The North/South and West/East Indicators.
- The Longitude Offset Indicator.
- The generic APRS digipeater path code.

Although the destination address appears to be quite unconventional, it is still a valid AX.25 address, consisting only of printable 7-bit ASCII values (shifted one bit left) — see the *Amateur Packet-Radio Link-Layer Protocol* specification for full details of the format of standard AX.25 addresses.

Information Field — This field contains the following data:

- The encoded longitude.
- The encoded course and speed.
- The display Symbol Code and Symbol Table Identifier.
- An optional field, containing either Mic-E telemetry data or a Mic-E status text string. The status string can contain plain text, Maidenhead

locator or the station's altitude.

Mic-E Destination Address Field

The standard AX.25 Destination Address field consists of 7 bytes, containing 6 callsign characters and the SSID (plus a number of other bits that are not of interest here). When used to carry Mic-E data, however, this field has a quite different format:

	Mic-E Data — DESTINATION ADDRESS FIELD Format						
	Lat Digit 1 + Message Bit A	***Lat Digit 2 + Message Bit B***	***Lat Digit 3 + Message Bit C***	***Lat Digit 4 + N/S Lat Indicator***	***Lat Digit 5 + Longitude Offset***	***Lat Digit 6 + W/E Long Indicator***	***APRS Digi Path Code***
Bytes:	1	1	1	1	1	1	1

The Destination Address field contains:

- Six encoded latitude digits specifying degrees (digits 1 and 2), minutes (digits 3 and 4) and hundredths of minutes (digits 5 and 6).
- 3-bit Mic-E message identifier (message bits A, B and C).
- North/South latitude indicator.
- Longitude offset (adds 0 degrees or 100 degrees to the longitude computation in the Information field).
- West/East longitude indicator.
- Generic APRS digipeater path (encoded in the SSID).

Destination Address Field Encoding

The table on the next page shows the encoding of the first 6 bytes of the Destination Address field, for all combinations of latitude digit, the 3-bit Mic-E message identifier (A/B/C), the latitude/longitude indicators and the longitude offset.

The encoding supports position ambiguity.

The ASCII characters shown in the table are left-shifted one bit position prior to transmission.

Mic-E Destination Address Field Encoding (Bytes 1–6)

Byte:	1-6	1-3	4	5	6
ASCII Char	***Lat Digit***	***Message A/B/C***	***N/S***	***Long Offset***	***W/E***
0	0	0	South	+0	East
1	1	0	South	+0	East
2	2	0	South	+0	East
3	3	0	South	+0	East
4	4	0	South	+0	East
5	5	0	South	+0	East
6	6	0	South	+0	East
7	7	0	South	+0	East
8	8	0	South	+0	East
9	9	0	South	+0	East
A	0	1 (Custom)			
B	1	1 (Custom)			
C	2	1 (Custom)			
D	3	1 (Custom)			
E	4	1 (Custom)			
F	5	1 (Custom)			
G	6	1 (Custom)			

Byte:	1-6	1-3	4	5	6
ASCII Char	***Lat Digit***	***Message A/B/C***	***N/S***	***Long Offset***	***W/E***
H	7	1 (Custom)			
I	8	1 (Custom)			
J	9	1 (Custom)			
K	space	1 (Custom)			
L	space	0	South	+0	East
P	0	1 (Std)	North	+100	West
Q	1	1 (Std)	North	+100	West
R	2	1 (Std)	North	+100	West
S	3	1 (Std)	North	+100	West
T	4	1 (Std)	North	+100	West
U	5	1 (Std)	North	+100	West
V	6	1 (Std)	North	+100	West
W	7	1 (Std)	North	+100	West
X	8	1 (Std)	North	+100	West
Y	9	1 (Std)	North	+100	West
Z	space	1 (Std)	North	+100	West

Note: the ASCII characters **A**–**K** are not used in address bytes 4–6.

For example, for a station at a latitude of 33 degrees 25.64 minutes north, in the western hemisphere, with longitude offset +0 degrees, and transmitting standard message identifier bits 1/0/0, the encoding of the first 6 bytes of the Destination Address field is as follows:

Destination Address Byte:	***1***	***2***	***3***	***4***	***5***	***6***
Latitude Digit	3	3	2	5	6	4
Message Bits	Message Bit A = 1 (Std)	Message Bit B = 0	Message Bit C = 0			
N/S Indicator				North		
Long Offset					+0	
W/E Indicator						West
Dest Address (ASCII Char)	**S**	**3**	**2**	**U**	**6**	**T**

Mic-E Messages

The first three bytes of the Destination Address field contain three message identifier bits: A, B and C. These bits allow one of 15 message types to be specified:

- 7 Standard messages
- 7 Custom messages
- 1 Emergency message

For the 7 Standard messages, one or more of the message identifier bits is a **1**, shown in the Mic-E Destination Address Field Encoding table as 1 (Std).

For the 7 Custom messages, one or more of the message identifier bits is a **1**, shown in the Mic-E Destination Address Field Encoding table as 1 (Custom).

For the Emergency message, all three message identifier bits are **0**.

The following table shows the encoding of Mic-E message types, for all combinations of the A/B/C message identifier bits:

Mic-E Message Types

A	*B*	*C*	*Standard Mic-E Message Type*	*Custom Mic-E Message Type*
1	1	1	M0: Off Duty	C0: Custom-0
1	1	0	M1: En Route	C1: Custom-1
1	0	1	M2: In Service	C2: Custom-2
1	0	0	M3: Returning	C3: Custom-3
0	1	1	M4: Committed	C4: Custom-4
0	1	0	M5: Special	C5: Custom-5
0	0	1	M6: Priority	C6: Custom-6
0	0	0	Emergency	

The Standard messages and the Emergency message have the same meaning for all APRS stations. The Custom messages may be assigned any arbitrary meaning.

Note: Support for Custom messages is optional. Original Mic-E units do not support Custom messages.

Note: If the A/B/C message identifier bits contain a mixture of Standard **1**s and Custom **1**s, the message type is “unknown”.

Some examples of message type encoding:

First 3 Destination Address Bytes	*Message Identifier Bits A/B/C*	*Message Type*	*Message*
S32	Standard 1 / 0 / 0	Standard	M3: Returning
F2D	Custom 1 / 0 / Custom 1	Custom	C2: Custom-2
234	0 / 0 / 0	Emergency	Emergency

Destination Address SSID Field

The SSID in the Destination Address field of a Mic-E packet is coded to specify either a conventional digipeater VIA path (contained in the Digipeater Addresses field of the AX.25 frame), or one of 15 generic APRS digipeater paths. See Chapter 4: APRS Data in the AX.25 Destination and Source Address Fields.

Mic-E Information Field

The Information field is used to complete the Position Report that was begun in the Destination Address field. The encoding used is different from the destination address since the content is not constrained to be printable, shifted 7-bit ASCII, as it is in the address. However, full 8-bit binary is not used — all values are offset by 28 and further operations (described below) are performed on some of the values to make almost all of the data printable ASCII.

The format of the Information field is as follows:

Mic-E Data — INFORMATION FIELD Format									
Data Type ID	*Longitude*			*Speed and Course*			*Symbol Code*	*Sym Table ID*	*Mic-E Telemetry Data*
	d+28	m+28	h+28	SP+28	DC+28	SE+28			*Mic-E Status Text*
Bytes: 1	1	1	1	1	1	1	1	1	n

Information Field Data

The first 9 bytes of the Information field contain the APRS Data Type Identifier, longitude, speed, course and symbol data.

The APRS Data Type Identifier is one of:

`` ` `` Current GPS data
(but not used in Kenwood TM-D700 radios) .

`'` Old GPS data
(or *Current* GPS data in Kenwood TM-D700 radios).

0x1c Current GPS data (Rev. 0 beta units only).

0x1d Old GPS data (Rev. 0 beta units only).

IMPORTANT NOTE: As explained in detail below, some of the bytes in the Information field are non-printing ASCII characters. In certain circumstances (such as incorrect TNC setup or inappropriate software), some of these non-printing characters may be dropped, making the Information field appear shorter than it really is. This will lead to incorrect decoding. (In particular, if the Information field appears to be less than 9 bytes long, the packet must be ignored).

Longitude Degrees Encoding

The **d+28** byte in the Information field contains the encoded value of the longitude degrees, in the range 0–179 degrees.

(Note that for longitude values in the range 0–9 degrees, the longitude offset is +100 degrees):

Mic-E Longitude Degrees Encoding

Long Deg	*ASCII Char*	*d+28*	*Long Offset*
0	v	118	**+100**
1	w	119	**+100**
2	x	120	**+100**
3	y	121	**+100**
4	z	122	**+100**
5	{	123	**+100**
6	|	124	**+100**
7	}	125	**+100**
8	~	126	**+100**
9	**DEL**	127	**+100**
10	&	38	+0
11	'	39	+0
12	(	40	+0
...			
97	}	125	+0
98	~	126	+0
99	**DEL**	127	+0

Long Deg	*ASCII Char*	*d+28*	*Long Offset*
100	ℓ	108	+100
101	m	109	+100
102	n	110	+100
103	o	111	+100
104	p	112	+100
105	q	113	+100
106	r	114	+100
107	s	115	+100
108	t	116	+100
109	u	117	+100
110	&	38	+100
111	'	39	+100
112	(	40	+100
...			
177	i	105	+100
178	j	106	+100
179	k	107	+100

Note from the table that the encoding is split into four separate pieces:

- 0–9 degrees: **d+28** is in the range 118–127 decimal, corresponding to the ASCII characters **v** to **DEL**.

  Important Note: The longitude offset is set to **+100 degrees** when the longitude is in the range 0–9 degrees.

- 10–99 degrees: **d+28** is in the range 38–127 decimal (corresponding to the ASCII characters **&** to **DEL**), and the longitude offset is +0 degrees.

- 100–109 degrees: **d+28** is in the range 108–117 decimal, corresponding to the ASCII characters **ℓ** (lower-case letter "L") to **DEL**, and the longitude offset is +100 degrees.
- 110–179 degrees: **d+28** is in the range 38–127 decimal (corresponding to the ASCII characters **&** to **DEL**), and the longitude offset is +100 degrees.

Thus the overall range of valid **d+28** values is 38–127 decimal (corresponding to ASCII characters **&** to **DEL**).

All of these characters (except **DEL**, for 9 and 99 degrees) are printable ASCII characters.

To decode the longitude degrees value:

1. subtract 28 from the **d+28** value to obtain **d**.
2. if the longitude offset is +100 degrees, add 100 to **d**.
3. subtract 80 if 180 ≤ **d** ≤ 189
   (i.e. the longitude is in the range 100–109 degrees).
4. or, subtract 190 if 190 ≤ **d** ≤ 199.
   (i.e. the longitude is in the range 0–9 degrees).

Longitude Minutes Encoding

The **m+28** byte in the Information field contains the encoded value of the longitude minutes, in the range 0–59 minutes:

Mic-E Longitude Minutes Encoding

Long Mins	*ASCII Char*	*m+28*
0	X	88
1	Y	89
2	Z	90
3	[	91
4	\	92
5	]	93
6	^	94
7	_	95
8	`	96
9	a	97

Long Mins	*ASCII Char*	*m+28*
10	&	38
11	'	39
12	(	40
13	)	41
14	*	42
...		
56	T	84
57	U	85
58	V	86
59	W	87

Note from the table that the encoding is split into two separate pieces:

- 0–9 minutes: **m+28** is in the range 88–97 decimal, corresponding to the ASCII characters **X** to **a**.
- 10–59 minutes: **m+28** is in the range 38–87 decimal (corresponding to the ASCII characters **&** to **W**).

Thus the overall range of valid **m+28** values is 38–97 decimal (corresponding

to ASCII characters **&** to **a**). All of these characters are printable ASCII characters.

To decode the longitude minutes value:

1. subtract 28 from the **m+28** value to obtain **m**.
2. subtract 60 if **m** ≥ 60.
   (i.e. the longitude minutes is in the range 0–9).

Longitude Hundredths of Minutes Encoding

The **h+28** byte in the Information field contains the encoded value of the longitude hundredths of minutes, in the range 0–99 minutes. This byte takes a value in the range 28 decimal (corresponding to 0 hundredths of a minute) through 127 decimal (corresponding to 99 hundredths of a minute).

To decode the longitude hundredths of minutes value, subtract 28 from the **h+28** value.

All of the possible values are printable ASCII characters (except 0–3 and 99 hundredths of a minute).

Speed and Course Encoding

The speed and course of a station are encoded in 3 bytes, designated **SP+28**, **DC+28** and **SE+28**.

The speed is in the range 0–799 knots, and the course is in the range 0–360 degrees (0 degrees represents an unknown or indefinite course, and 360 degrees represents due north).

The encoded speed and course are spread over the three bytes, as follows:

Speed		***Course***
Encoded Speed (hundreds/tens of knots)	Encoded Speed (units) and Encoded Course (hundreds of degrees)	Encoded Course (tens/units)
SP+28	**DC+28**	**SE+28**

SP+28 Encoding

The **SP+28** byte contains the encoded speed, in hundreds/tens of knots, according to this table:

SP+28 Speed Encoding (hundreds/tens of knots)

Speed knots	ASCII Char		SP +28	
0-9	ℓ	0x1c	108	28
10-19	m	0x1d	109	29
20-29	n	0x1e	110	30
30-39	o	0x1f	111	31
40-49	p	␣	112	32
50-59	q	!	113	33
60-69	r	"	114	34
70-79	s	#	115	35
80-89	t	$	116	36
90-99	u	%	117	37
100-109	v	&	118	38
110-119	w	'	119	39
120-129	x	(	120	40
130-139	y	)	121	41
140-149	z	*	122	42
150-159	{	+	123	43
160-169	|	,	124	44
170-179	}	-	125	45
180-189	~	.	126	46
190-199	DEL	/	127	47

Speed knots	ASCII Char	SP +28
200-209	0	48
210-219	1	49
220-229	2	50
230-239	3	51
240-249	4	52
250-259	5	53
260-269	6	54
270-279	7	55
280-289	8	56
290-299	9	57
300-310	:	58
310-320	;	59
...		
730-739	e	101
740-749	f	102
750-759	g	103
760-769	h	104
770-779	i	105
780-789	j	106
790-799	k	107

Note: The ASCII characters shown in white on a black background are non-printing characters.

Note: For speeds in the range 0–199 knots, there are two encoding schemes in existence. Hence there are two columns for the ASCII character, and two columns for the corresponding **SP+28** byte values.

For example, for a speed of 73 knots (i.e. in the range 70–79), the **SP+28** byte may contain either **s** or **#**, depending on the encoding method used. Both are equally valid.

The decoding algorithm described later handles either of these encoding schemes.

DC+28 Encoding

The **DC+28** byte contains the encoded units of speed, plus the encoded course in hundreds of degrees:

DC+28 Speed / Course Encoding (units of knots/hundreds of degrees)

Knots (units)	Course (deg)	ASCII Char		DC +28	
0	0-99	␣	0x1c	32	28
0	100-199	!	0x1d	33	29
0	200-299	"	0x1e	34	30
0	300-360	#	0x1f	35	31
1	0-99	*	&	42	38
1	100-199	+	'	43	39
1	200-299	,	(	44	40
1	300-360	-	)	45	41
2	0-99	4	0	52	48
2	100-199	5	1	53	49
2	200-299	6	2	54	50
2	300-360	7	3	55	51
3	0-99	>	:	62	58
3	100-199	?	;	63	59
3	200-299	@	<	64	60
3	300-360	A	=	65	61
4	0-99	H	D	72	68
4	100-199	I	E	73	69
4	200-299	J	F	74	70
4	300-360	K	G	75	71

Knots (units)	Course (deg)	ASCII Char		DC +28	
5	0-99	R	N	82	78
5	100-199	S	O	83	79
5	200-299	T	P	84	80
5	300-360	U	Q	85	81
6	0-99	\	X	92	88
6	100-199	]	Y	93	89
6	200-299	^	Z	94	90
6	300-360	_	[	95	91
7	0-99	f	b	102	98
7	100-199	g	c	103	99
7	200-299	h	d	104	100
7	300-360	i	e	105	101
8	0-99	p	ℓ	112	108
8	100-199	q	m	113	109
8	200-299	r	n	114	110
8	300-360	s	o	115	111
9	0-99	z	v	122	118
9	100-199	{	w	123	119
9	200-299	|	x	124	120
9	300-360	}	y	125	121

Note: The ASCII characters shown in white on a black background are non-printing characters.

Note: There are two encoding schemes in existence for the **DC+28** byte. Hence there are two columns for the ASCII character, and two columns for the corresponding **DC+28** byte values.

For example, for a speed of 73 knots (i.e. units=3) and a bearing of 294 degrees (i.e. in the range 200–299), the **DC+28** byte may contain either **@** or **<**, depending on the encoding method used. Both are equally valid.

The decoding algorithm described later handles either of these encoding schemes.

SE+28 Encoding

The **SE+28** byte contains the encoded tens and units of degrees of the course:

SE+28 Course Encoding (tens/units of degrees)

Course (deg)	*ASCII Char*	*m+28*
0	**0x1c**	28
1	**0x1d**	29
2	**0x1e**	30
3	**0x1f**	31
4	␣	32
5	!	33
6	"	34
7	#	35
8	$	36
9	%	37
10	&	38
11	'	39
12	(	40
13	)	41
14	*	42

Long Mins	*ASCII Char*	*m+28*
15	+	43
16	,	44
17	-	45
18	.	46
19	/	47
...		
91	w	119
92	x	120
93	y	121
94	z	122
95	{	123
96	|	124
97	}	125
98	~	126
99	**DEL**	127

Example of Mic-E Speed and Course Encoding

For a speed of 86 knots and a course of 194 degrees, the encoding is:

SP+28: The speed is in the range 80–89 knots. From the **SP+28** encoding table, the **SP+28** byte may be either **t** or **$**.

DC+28: The units of speed are 6, and the course is in the range 100–199 degrees. From the **DC+28** encoding table, the **DC+28** byte may be either **]** or **Y**.

SE+28: The course in tens and units of degrees is 94. From the **SE+28** encoding table, the **SE+28** byte will be **z**.

Decoding the Speed and Course

To decode the speed and course:

SP+28: To obtain the speed in tens of knots, subtract 28 from the **SP+28** value and multiply by 10.

DC+28: Subtract 28 from the **DC+28** value and divide the result by 10. The quotient is the units of speed. The remainder is the course in hundreds of degrees.

SE+28: To obtain the tens and units of degrees, subtract 28 from the **SE+28** value.

Finally, make these speed and course adjustments:

- If the computed speed is ≥ 800 knots, subtract 800.
- If the computed course is ≥ 400 degrees, subtract 400.

Example of Decoding the Information Field Data

If the first 9 bytes of the Information field contain `` `(_fn"Oj/ ``, and the destination address specifies that the station is in the western hemisphere with a longitude offset of +100 degrees, then the data is decoded as follows:

- `` ` `` is the APRS Data Type Identifier for a Mic-E packet containing current GPS data.
- `(` is the **d+28** byte. The `(` character has the value 40 decimal. Subtracting 28 gives 12. The longitude offset (in the destination address) is +100 degrees, so the longitude is 100 + 12 = 112 degrees.
- `_` is the **m+28** byte. The `_` character has the value 95 decimal. Subtracting 28 gives 67. This is ≥ 60, so subtracting 60 gives a value of 7 minutes longitude.
- `f` is the **h+28** byte. The `f` character has the value 102 decimal. Subtracting 28 gives 74 hundredths of a minute.

Thus the longitude is 112 degrees 7.74 minutes west.

The speed and course are calculated as follows:

- `n` is the **SP+28** byte. The `n` character has the value 110 decimal. After subtracting 28, the result is 82. As this is ≥ 80, a further 80 is subtracted, leaving a result of 2 tens of knots.
- `"` is the **DC+28** byte. The `"` character has the value 34 decimal. Subtracting 28 gives 6. Dividing this by 10 gives a quotient of 0 (units of speed). Adding the first part of the speed, multiplied by 10 (i.e. 20) to the quotient (0) gives a final computed speed of 20 knots.
  The remainder from the division is 6. Subtracting 4 gives the course in hundreds of degrees; i.e. 2.
- `O` (upper-case letter "O") is the **SE+28** byte. The `O` character has the value 79 decimal. Subtracting 28 gives 51. Adding this to the remainder calculated above, multiplied by 100 (i.e. 200), gives the final value of 251 degrees for the course.

The last two characters (`j/`) represent the jeep symbol from the Primary Symbol Table.

Mic-E Position Ambiguity

As mentioned in Chapter 6 (Time and Position Formats), a station may reduce the precision of its position by introducing position ambiguity. This is also possible in Mic-E data format.

The position ambiguity is specified for the latitude (in the destination address). The same degree of ambiguity will then also apply to the longitude.

For example, if the destination address is `T4SQZZ`, the last two digits of the

latitude are ambiguous (represented by **ZZ**). Then, if the longitude data in the Information field is **(_f**, as in the above example, the last two digits of the computed longitude will be ignored — that is, the longitude will be 112 degrees 7 minutes.

Mic-E Telemetry Data

The Information field may optionally contain either Mic-E telemetry data values or Mic-E status text.

If the byte following the Symbol Table Identifier is one of the Telemetry Flag characters (**`**,**'** or 0x1d), then telemetry data follows:

Optional Mic-E Telemetry Data					
Telemetry Flag	***Telemetry Data Channels***				
F	Ch 1	Ch 2	Ch 3	Ch 4	Ch 5
Bytes: 1	1/2	1/2	1/2	1/2	1/2

The Telemetry Flag F is one of:

`	2 printable hex telemetry values follow (channels 1 and 3).
'	5 printable hex telemetry values follow.
0x1d	5 binary telemetry values follow (Rev. 0 beta units only).

If F is **`** or **'**, each channel requires 2 bytes, containing a 2-digit printable hexadecimal representation of a value ranging from 0–255. For example, 254 is represented as **FE**.

If F is 0x1d, each channel requires one byte, containing an 8-bit binary value.

For example, if the telemetry data is **'7200007100**, the **'** indicates that 5 bytes of telemetry follow, coded in hexadecimal:

0x72 = 114 decimal
0x00 = 0 decimal
0x00 = 0 decimal
0x71 = 113 decimal
0x00 = 0 decimal

Mic-E Status Text

As an alternative to telemetry data, the packet may include Mic-E status text. The status text may be any length that fits in the rest of the Information field.

The Mic-E status text must not start with **`**,**'** or 0x1d, otherwise it will be confused with telemetry data.

It is possible to include a standard APRS-formatted position in the Mic-E status text field. A suitable position will cause the APRS display software to override any position data the Mic-E has encoded. This is useful if using a Mic-E without a GPS receiver.

Note: The Kenwood radios automatically insert a special type code at the front of the status text string (i.e. in the 10th character of the Information field):

Kenwood TH-D7: **>**
Kenwood TM-D700: **]**

These characters should not be confused with the APRS Data Type Identifier that appears at the start of reports.

It is envisaged that other Mic-E-compatible devices will be allocated their own type codes in future.

Note: When Kenwood radios receive the status, they can only display a small number of text characters:

Kenwood TH-D7: 20 characters
Kenwood TM-D700: 28 characters

Note: The Kenwood TM-D700 radio uses the **'** (apostrophe) instead of the **`** (grave) APRS Data Type Identifier to represent current GPS data. A suggested way of detecting this situation is to examine the first and 10th characters of the Information field; if they are **'** and **]** respectively, then the packet is almost certainly from a TM-D700.

Maidenhead Locator in the Mic-E Status Text Field

The Mic-E status text field can contain a Maidenhead locator.

If the locator is followed by a plain text comment, the first character of the text *must* be a space. For example:

```
IO91SX/G␣Hello␣world    (from a Mic-E or PIC-E)
>IO91SX/G␣Hello␣world   (from a Kenwood TH-D7)
]IO91SX/G␣Hello␣world   (from a Kenwood TM-D700)
```

(**/G** is the grid locator symbol).

Altitude in the Mic-E Status Text Field

The Mic-E status text field can contain the station's altitude. The altitude is expressed in the form `xxx`**}**, where `xxx` is in meters relative to 10km below mean sea level (the deepest ocean), to base 91.

For example, to compute the `xxx` characters for an altitude of 200 feet:

200 feet = 61 meters = 10061 meters relative to the datum
$10061 / 91^2$ = **1**, remainder 1780
1780 / 91 = **19**, remainder **51**

Adding 33 to each of the highlighted values gives 34, 52 and 84 for the ASCII codes of xxx.

Thus the 4-character altitude string is **"4T}**

Some examples:

```
"4T}
>"4T}
]"4T}
```

Mic-E Data in Non-APRS Networks

Some parts of the Mic-E AX.25 Information field may contain binary data (i.e. non-printable ASCII characters). If such a packet is constrained to the APRS network, this should not cause any difficulties.

If, however, the packet is to be forwarded via a network that does not reliably preserve binary data (e.g. the Internet), then it is necessary to convert the data to a format that will preserve it.

Further, if the packet subsequently re-emerges back onto the APRS network, it will then be necessary to re-convert the data back to its original format.

11 OBJECT AND ITEM REPORTS

Objects and Items

Any APRS station can manually report the position of an APRS entity (e.g. another station or a weather phenomenon). This is intended for situations where the entity is not capable of reporting its own position.

APRS provides two types of report to support this:

- Object Reports
- Item Reports

Object Reports specify an Object's position, can have an optional timestamp, and can include course/speed information or other Extended Data. Object Reports are intended primarily for plotting the positions of moving objects (e.g. spacecraft, storms, marathon runners without trackers).

Item Reports specify an Item's position, but cannot have a timestamp. While Item reports may also include course/speed or other Extended Data, they are really intended for inanimate things that are occasionally posted on a map (e.g. marathon checkpoints or first-aid posts). Otherwise they are handled in the same way as Object Reports.

Objects are distinguished from each other by having different Object names. Similarly, Items are distinguished from each other by having different Item names.

Implementation Recommendation: When an APRS Object/Item is displayed on the screen, the callsign of the station sending the report should be associated with the Object/Item.

Replacing an Object / Item

A fundamental precept of APRS is that any station may take over the reporting responsibility for an APRS Object or Item, by simply transmitting a new report with the same Object/Item name.

The replacement report may specify the existing location or a new location.

The original station will cease transmitting an Object/Item Report when it sees an incoming report with the same name from another station.

Killing an Object / Item

To kill an Object/Item, a station transmits a new Object/Item Report, with a "kill" character following the Object/Item name.

Implementation Recommendation: When an Object/Item is killed it should be removed from display on the screen. However, the data associated with the Object/Item should be retained internally in case it is needed later.

Object Report Format

An Object Report has a *fixed* 9-character Object name, which may consist of any printable ASCII characters.

Object names are case-sensitive.

The ; is the APRS Data Type Identifier for an Object Report, and a * or _ separates the Object name from the rest of the report:

* indicates a live Object.

_ indicates a killed Object.

The position may be in lat/long or compressed lat/long format, and the report may also contain Extended Data.

An Object always has a timestamp.

The Comment field may contain any appropriate APRS data (see the *Comment Field* section in Chapter 5: APRS Data in the AX.25 Information Field).

Object Report Format — with Lat/Long position

	;	Object Name	* or _	Time DHM / HMS	Lat	Sym Table ID	Long	Symbol Code	Course/Speed Power/Height/Gain/Dir Radio Range DF Signal Strength Area Object	Comment (max 36 chars with Data Extension, or 43 without)
Bytes:	1	9	1	7	8	1	9	1	7	0-36/43

Examples

`;LEADER___*092345z4903.50N/07201.75W>088/036` A live Object. At 2345 hours zulu on the 9th of the month, the "Leader" was in the car at 49°3'30"N/72°1'45"W, heading 88 deg at 36 knots.

`;LEADER___*092345z4903.50N/07201.75W>088/036` The same Object, now killed.

Object Report Format — with Compressed Lat/Long position

	;	Object Name	* or _	Time DHM / HMS	Compressed Position Data /YYYYXXXX$csT	Comment
Bytes:	1	9	1	7	13	43

Example

`;LEADER___*092345z/5L!!<*e7>7P[` The "Leader" was in the car at 49°30'00"N/ 72°45'00"W, heading 88 deg at 36.2 knots.

Item Report Format

An Item Report has a *variable-length* Item name, 3–9 characters long. The name may consist of any printable ASCII characters *except* **!** or **_**.

Item names are case-sensitive.

The **)** is the APRS Data Type Identifier for an Item Report, and a **!** or **_** separates the Item name from the rest of the report:

! indicates a live Item.

_ is the Item "kill" character.

The position may be in lat/long or compressed lat/long format. There is no provision for a timestamp. The report may also contain Extended Data.

The Comment field may contain any appropriate APRS data (see the *Comment Field* section in Chapter 5: APRS Data in the AX.25 Information Field).

Item Report Format — with Lat/Long position

	)	*Item Name*	**!** or **_**	*Lat*	*Sym Table ID*	*Long*	*Symbol Code*	*Course/Speed* *Power/Height/Gain/Dir* *Radio Range* *DF Signal Strength* *Area Object*	*Comment (max 36 chars with Data Extension, or 43 without)*
Bytes:	1	3-9	1	8	1	9	1	7	0-36/43

Examples

`)AID␣#2!4903.50N/07201.75WA`	First Aid Station #2 is at 49°3'30"N/72°1'45"W. (**/A** is the symbol for Aid Station).
`)G/WB4APR!53␣␣.␣␣N\002␣␣.␣␣Wd`	A rare DX station "somewhere in England". (**\d** is the symbol for DX Spot).
`)AID␣#2_4903.50N/07201.75WA`	The First Aid Station has closed down.

Item Report Format — with Compressed Lat/Long position

	)	*Item Name*	**!** or **_**	*Compressed Position Data* `/YYYYXXXX$csT`	*Comment*
Bytes:	1	3-9	1	13	43

Example

`)MOBIL!\5L!!<*e79␣sT`	Mobil Gas Station is at 49°30'00"N/72°45'00"W. (**\9** is the symbol for Gas Station).

Area Objects

Using the \ℓ symbol (i.e. the lower-case letter "L" symbol from the Alternate Symbol Table) it is possible to define circle, line, ellipse, triangle and box objects in all colors, either open or filled in, any size from 60 feet to 100 miles.

These Objects are useful for real-time events such as for a search-and-rescue, or adding a special road or route for a special event.

The Object format is specified as a 7-character APRS Data Extension `Tyy/Cxx` immediately following the ℓ Symbol Code. For example:

`;OBJECT␣␣␣*ddmm.hhN\dddmm.hhWℓTyy/Cxx`

where:

`T` is the type of object shape.

`/C` is the color of the object.

`yy` is the square root of the latitude offset in 1/100ths of a degree.

`xx` is the square root of the longitude offset in 1/100ths of a degree.

The object type and color codes are as follows:

T	*Object Type*
0	Open circle
1	Line (offset: down/right)
2	Open ellipse
3	Open triangle
4	Open box
5	Color-filled circle
6	Line (offset: down/left)
7	Color-filled ellipse
8	Color-filled triangle
9	Color-filled box

/C	*Object Color*	*Intensity*
/0	Black	High
/1	Blue	High
/2	Green	High
/3	Cyan	High
/4	Red	High
/5	Violet	High
/6	Yellow	High
/7	Gray	High
/8	Black	Low
/9	Blue	Low
10	Green	Low
11	Cyan	Low
12	Red	Low
13	Violet	Low
14	Yellow	Low
15	Gray	Low

The latitude/longitude position is the upper left corner of the object, and the offsets are relative to this position — the yy offset is *down* from this position and the xx offset is to the *right* of this position. (An exception is the special case of a Type 6 line which is drawn down and to the *left*).

Here are some examples of Object Position Reports. The latitude and longitude offsets are each one degree (i.e. 100/100ths of a degree), so yy = xx = √100 = 10.

```
;SEARCH___*092345z4903.50N\07201.75Wℓ710/310
```
A high intensity cyan filled ellipse, yy=10, xx=10

```
;SEARCH___*092345z4903.50N\07201.75Wℓ8101310
```
A low intensity violet filled triangle, yy=10, xx=10

Further, with the line option (Type 1 and Type 6) it is possible to specify a "corridor" either side of the central line. The width of the corridor (in miles) either side of the line is specified in the comment text, enclosed by **{}**.

For example:

```
;FLIGHTPTH*4903.50N\07201.75Wℓ610/310{100}
```
A high intensity cyan line, with a 100-mile corridor either side

Note: The color fill option should be used with care, since a color-filled object will obscure information displayed underneath it.

Signpost Objects/Items

Signpost Objects/Items (with the symbol **\m**) display as a yellow box with a 1–3-character overlay on them. The overlay is specified by enclosing the 1–3 characters in braces in the comment field. Thus a signpost with {55} would appear as a sign with **55** on it.

For example:

```
)I91_3N!4903.50N\07201.75Wm{55}
```

This was originally designed for posting the speed of traffic past speed measuring devices, but can be used for any purpose.

Implementation Recommendation: Signposts should not display any callsign or name, and to avoid clutter should only be displayed at close range.

Obsolete Object Format

Some stations transmit Object reports without the **;** APRS Data Type Identifier. This format is obsolete. Some software may still decode such data as an Object, but it should now be interpreted as a Status Report.

12 WEATHER REPORTS

Weather Report Types

APRS is an ideal tool for reporting weather conditions via packet. APRS supports serial data transmissions from the Peet Brothers, Ultimeter and Davis home weather stations. It is even possible to mount an Ultimeter remotely with only a TNC and radio to report and plot conditions. APRS is also ideally suited for the Skywarn weather observer initiative.

APRS supports three types of Weather Report:

- Raw Weather Report
- Positionless Weather Report
- Complete Weather Report

Data Type Identifiers

The following APRS Data Type Identifiers are used in Weather Reports containing raw data:

!	Ultimeter 2000
#	Peet Bros U-II
$	Ultimeter 2000
*****	Peet Bros U-II
_	Positionless weather data

In addition, where the raw data has been post-processed (for example, by the insertion of station location information), the four position Data Type Identifiers **!**, **=**, **/** and **@** may be used instead. In this case, the Weather Report is identified with the weather symbol **/_** or **_** in the APRS Data.

Raw Weather Reports

Raw weather data from a stand-alone weather station is contained in the Information Field of an APRS AX.25 frame:

	Raw Weather Report Format	
	! *or* ***#*** *or* ***$*** *or* *******	***Raw Weather Data***
Bytes:	1	n

Examples

```
!!006B005803500000----03E9--------002105140000005D          Ultimeter 2000
#50B7500820082                                              Peet Bros U-II
$ULTW0031003702CE0069----000086A00001----011901CC00000005   Ultimeter 2000
*7007600000000                                              Peet Bros U-II
```

Positionless Weather Reports

Generic raw weather data from a stand-alone weather station is contained in the Information Field of an APRS AX.25 frame:

Positionless Weather Report Format				
_	***Time MDHM***	***Positionless Weather Data***	***APRS Software*** S	***WX Unit*** uuuu
Bytes: 1	8	n	1	2-4
Example _10090556c220s004g005t077r000p000P000h50b09900wRSW report derived from Radio Shack WX station data.				

APRS Software Type

A Weather Report may contain a single-character code `S` for the type of APRS software that is running at the weather station:

- **d** = APRSdos
- **M** = MacAPRS
- **P** = pocketAPRS
- **S** = APRS+SA
- **W** = WinAPRS
- **X** = X-APRS (Linux)

Weather Unit Type

A Weather Report may contain a 2–4 character code `uuuu` for the type of weather station unit. The following codes have been allocated:

- **Dvs** = Davis
- **HKT** = Heathkit
- **PIC** = PIC device
- **RSW** = Radio Shack
- **U-II** = Original Ultimeter U-II (auto mode)
- **U2R** = Original Ultimeter U-II (remote mode)
- **U2k** = Ultimeter 500/2000
- **U2kr** = Remote Ultimeter logger
- **U5** = Ultimeter 500
- **Upkm** = Remote Ultimeter packet mode

Users may specify any other 2–4 character code for devices not in this list.

Positionless Weather Data

The format of weather data within a Positionless Weather Report differs according to the type of weather station unit, but generically consists of some or all of the following elements:

	Positionless Weather Data								
	Wind Direction **c**ccc	***Wind Speed*** **s**sss	***Gust*** **g**ggg	***Temp*** **t**ttt	***Rain Last Hr*** **r**rrr	***Rain Last 24 Hrs*** **p**ppp	***Rain Since Midnight*** **P**PPP	***Humidity*** **h**hh	***Barometric Pressure*** **b**bbbbb
Bytes:	4	4	4	4	4	4	4	3	5

where: **c** = wind direction (in degrees).

s = sustained one-minute wind speed (in mph).

g = gust (peak wind speed in mph in the last 5 minutes).

t = temperature (in degrees Fahrenheit). Temperatures below zero are expressed as -01 to -99.

r = rainfall (in hundredths of an inch) in the last hour.

p = rainfall (in hundredths of an inch) in the last 24 hours.

P = rainfall (in hundredths of an inch) since midnight.

h = humidity (in %. `00` = 100%).

b = barometric pressure (in tenths of millibars/tenths of hPascal).

Other parameters that are available on some weather station units include:

L = luminosity (in watts per square meter) 999 and below.

ℓ (lower-case letter "L") = luminosity (in watts per square meter) 1000 and above.
(L is inserted in place of one of the rain values).

s = snowfall (in inches) in the last 24 hours.

= raw rain counter

Note: The weather report must include at least the MDHM date/timestamp, wind direction, wind speed, gust and temperature, but the remaining parameters may be in a different order (or may not even exist).

Note: Where an item of weather data is unknown or irrelevant, its value may be expressed as a series of dots or spaces. For example, if there is no wind speed/direction/gust sensor, the wind values could be expressed as:

c...s...g... or **c___s___g___**

For example, Jim's rain gauge may produce a report like this:

```
_10090556c...s...g...t...P012Jim
```

(The date/timestamp, wind direction/speed/gust and temperature parameters must be included, even though they are not meaningful).

Location of a Raw and Positionless Weather Stations

APRS cannot display weather data on a map until it knows the location of the sending station. In the case of a station transmitting Raw or Positionless Weather Reports, the station has to occasionally send an additional packet containing its position (using any of the legal lat/long and compressed lat/long position formats described earlier).

Symbols with Raw and Positionless Weather Stations

Because Raw and Positionless Weather Reports do not contain a display symbol in the AX.25 Information field, it is possible to specify the symbol in a generic APRS destination address (e.g. `GPSHW` or `GPSE63`) instead. Alternatively, if the weather station is on a balloon, the SSID –11 may be used in the source address (e.g. `N0QBF-11`).

See Chapter 20: APRS Symbols for more detail on the usage of symbols.

Complete Weather Reports with Timestamp and Position

An APRS Complete Weather Report can contain a timestamp and location information, using any of the legal lat/long and compressed lat/long position formats described earlier. An APRS Object may also have weather information associated with it.

Examples of report formats are shown below. Note that the Symbol Code in every case is the **_** (underscore). Also, the 7-byte Wind Direction and Wind Speed Data Extension replace the **c**`ccc` and **s**`sss` fields of a Positionless Weather Report.

	Complete Weather Report Format — with Lat/Long position, no Timestamp								
	! *or* ***=***	***Lat***	***Sym Table ID***	***Long***	***Symbol Code*** **_**	***Wind Directn/ Speed***	***Weather Data***	***APRS Software*** S	***WX Unit*** uuuu
Bytes:	1	8	1	9	1	7	n	1	2-4

Examples

```
!4903.50N/07201.75W_220/004g005t077r000p000P000h50b09900wRSW
!4903.50N/07201.75W_220/004g005t077r000p000P000h50b.....wRSW
```

Complete Weather Report Format — with Lat/Long position and Timestamp

	/ or @	Time DHM / HMS	Lat	Sym Table ID	Long	Symbol Code _	Wind Directn/ Speed	Weather Data	APRS Software S	WX Unit uuuu
Bytes:	1	7	8	1	9	1	7	n	1	2-4

Example

```
@092345z4903.50N/07201.75W_220/004g005t-07r000p000P000h50b09900wRSW
```

Complete Weather Report Format — with Compressed Lat/Long position, no Timestamp

	! or =	Sym Table ID	Comp Lat YYYY	Comp Long XXXX	Symbol Code _	Comp Wind Directn/ Speed	Comp Type T	Weather Data	APRS Software S	WX Unit uuuu
Bytes:	1	1	4	4	1	2	1	n	1	2-4

Example

```
=/5L!!<*e7> _7P[g005t077r000p000P000h50b09900wRSW
```

Complete Weather Report Format — with Compressed Lat/Long position, with Timestamp

	/ or @	Time DHM / HMS	Sym Table ID	Comp Lat YYYY	Comp Long XXXX	Symbol Code _	Comp Wind Directn/ Speed	Comp Type T	Weather Data	APRS Software S	WX Unit uuuu
Bytes:	1	7	1	4	4	1	2	1	n	1	2-4

Example

```
@092345z/5L!!<*e7 _7P[g005t077r000p000P000h50b09900wRSW
```

Complete Weather Report Format — with Object and Lat/Long position

	;	Object Name	*	Time DHM / HMS	Lat	Sym Table ID	Long	Symbol Code _	Wind Directn/ Speed	Weather Data	APRS Software S	WX Unit uuuu
Bytes:	1	9	1	7	8	1	9	1	7	n	1	2-4

Examples

```
;BRENDA___*4903.50N/07201.75W_220/004g005t077r000p000P000h50b09900wRSW
;BRENDA___*092345z4903.50N/07201.75W_220/004g005b0990
```

Storm Data

APRS reports can contain data relating to tropical storms, hurricanes and tropical depressions. The format of the data is as follows:

	Storm Data									
	Direction	***/***	***Speed***	***Storm Type*** /ST	***Sustained Wind Speed*** /www	***Peak Wind Gusts*** ^GGG	***Central Pressure*** /pppp	***Radius Hurricane Winds*** >RRR	***Radius Tropical Storm Winds*** &rrr	***Radius Whole Gale*** %ggg
Bytes:	3	1	3	3	4	4	5	4	4	4

where: ST = **TS** (Tropical Storm)
HC (Hurricane)
TD (Tropical Depression).

www = sustained wind speed (in knots).

GGG = gust (peak wind speed in knots).

pppp = central pressure (in millibars/hPascal)

RRR = radius of hurricane winds (in nautical miles).

rrr = radius of tropical storm winds (in nautical miles).

ggg = radius of "whole gale" (i.e. 50 knot) winds (in nautical miles). Optional.

Storm data will usually be included in an Object Report, but may also be included in a Position Report or an Item Report.

The display symbol will be either:

\@	Hurricane/Tropical Storm (current position)
/@	Hurricane (predicted future position)

For example, the progress of Hurricane Brenda could be expressed in Object Reports like these:

```
;BRENDA___*092345z4903.50N\07202.75W@088/036/HC/150^200/0980>090&030%040
;BRENDA___*100045z4905.50N/07201.75W@101/047/HC/104^123/0980>065&020%040
```

National Weather Service Bulletins

APRS supports the dissemination of National Weather Service bulletins. See Chapter 14: Messages, Bulletins and Announcements.

13 TELEMETRY DATA

Telemetry Report Format

The AX.25 Information field can contain telemetry data. The APRS Data Type Identifier is **T**.

The report Sequence Number is a 3-character value — typically a 3-digit number, or the three letters **MIC**. In the case of **MIC**, there may or may not be a comma preceding the first analog data value.

There are five 8-bit unsigned analog data values (expressed as 3-digit decimal numbers in the range 000–255), followed by a single 8-bit digital data value (expressed as 8 bytes, each containing **1** or **0**).

The Kantronics KPC-3+ TNC and APRS Micro Interface Module (MIM) use this format.

Telemetry Report Format								
T	*Sequence No* #xxx,	*Analog Value 1* aaa,	*Analog Value 2* aaa,	*Analog Value 3* aaa,	*Analog Value 4* aaa,	*Analog Value 5* aaa,	*Digital Value* bbbbbbbb	*Comment*
Bytes: 1	5	4	4	4	4	4	8	n

Examples
```
T#005,199,000,255,073,123,01101001
T#MIC199,000,255,073,123,01101001
```

On-Air Definition of Telemetry Parameters

In principle, received telemetry data may be interpreted in any appropriate way. In practice, however, an APRS user can define the telemetry parameters (such as quadratic coefficients for the analog values, or the meaning of the binary data) at any time, and then send these definitions as APRS messages. Other stations receiving these messages will then know how to interpret the data.

This is achieved by sending four messages:

- A Parameter Name message.
- A Unit/Label message.
- An Equation Coefficients message.
- A Bit Sense/Project Name message.

The messages addressee is the callsign of the station transmitting the telemetry data. For example, if N0QBF launches a balloon with the callsign N0QBF-11, then the four messages are addressed to N0QBF-11.

See Chapter 14: Messages, Bulletins and Announcements for full details of message formats.

Parameter Name Message

The Parameter Name message contains the names (N) associated with the five analog channels and the 8 digital channels. Its format is as follows:

Telemetry Parameter Name Message Data
Note the different byte counts, which include commas where shown. The list may stop at any field.

	PARM.	***A1*** N...	***A2*** ,N...	***A3*** ,N...	***A4*** ,N...	***A5*** ,N...	***B1*** ,N...	***B2*** ,N...	***B3*** ,N...	***B4*** ,N...	***B5*** ,N...	***B6*** ,N...	***B7*** ,N...	***B8*** ,N...
Bytes:	5	1-7	1-7	1-6	1-6	1-5	1-6	1-5	1-4	1-4	1-4	1-3	1-3	1-3

Example

```
:N0QBF-11␣:PARM.Battery,Btemp,ATemp,Pres,Alt,Camra,Chut,Sun,10m,ATV
```

Note: The field widths are not all the same (this is a legacy arising from earlier limitations in display screen width). Note also that the byte counts *include* the comma separators where shown.

The list can terminate after any field.

Unit/Label Message

The Unit/Label message specifies the units (U) for the analog values, and the labels (L) associated with the digital channels:

Telemetry Unit/Label Message Data
Note the different byte counts, which include commas where shown. The list may stop at any field.

	UNIT.	***A1*** U...	***A2*** ,U...	***A3*** ,U...	***A4*** ,U...	***A5*** ,U...	***B1*** ,L...	***B2*** ,L...	***B3*** ,L...	***B4*** ,L...	***B5*** ,L...	***B6*** ,L...	***B7*** ,L...	***B8*** ,L...
Bytes:	5	1-7	1-7	1-6	1-6	1-5	1-6	1-5	1-4	1-4	1-4	1-3	1-3	1-3

Example

```
:N0QBF-11␣:UNIT.v/100,deg.F,deg.F,Mbar,Kft,Click,OPEN,on,on,hi
```

Note: Again, the field widths are not all the same, and the byte counts *include* the comma separators where shown.

The list can terminate after any field.

Equation Coefficients Message

The Equation Coefficients message contains three coefficients (a, b and c) for each of the five analog channels.

Telemetry Equation Coefficients Message Data
The list may stop at any field. Value = $a \times v^2 + b \times v + c$

		A1			A2			A3			A4			A5		
	EQNS.	a	,b	,c	,a	,b	,c	,a	,b	,c	,a	,b	,c	,a	,b	,c
Bytes:	5	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

Example
```
:N0QBF-11␣:EQNS.0,5.2,0,0,.53,-32,3,4.39,49,-32,3,18,1,2,3
```

To obtain the final value of an analog channel, these coefficients are substituted into the equation:

$$a \times v^2 + b \times v + c$$

where v is the raw received analog value.

For example, analog channel A1 in the above beacon examples relates to the battery voltage, expressed in hundredths of volts, and a = 0, b = 5.2, c = 0. If the raw received value v is 199, then the voltage is calculated as:

$$\begin{aligned} \text{voltage} &= 0 \times 199^2 + 5.2 \times 199 + 0 \\ &= 1034.8 \text{ hundredths of a volt} \\ &= 10.348 \text{ volts} \end{aligned}$$

Bit Sense/ Project Name Message

The Bit Sense/Project Name message contains two types of information:

- An 8-bit pattern of ones and zeros, specifying the sense of each digital channel that matches the corresponding label.
- The name of the project associated with the telemetry station.

Telemetry Bit Sense/Project Name Message Data

	BITS.	B1 x	B2 x	B3 x	B4 x	B5 x	B6 x	B7 x	B8 x	Project Title
Bytes:	5	1	1	1	1	1	1	1	1	0-23

Example
```
:N0QBF-11␣:BITS.10110000,N0QBF's Big Balloon
```

Thus in the above message examples, if digital channel B1 is 1, this indicates the camera has clicked. If channel B2 is 0, the parachute has opened, and so on.

14 MESSAGES, BULLETINS AND ANNOUNCEMENTS

APRS messages, bulletins and announcements are packets containing free format text strings, and are intended to convey human-readable information. A message is intended for reception by a single specified recipient, and an acknowledgement is usually expected. Bulletins and announcements are intended for reception by multiple recipients, and are not acknowledged.

Messages

An APRS message is a text string with a specified addressee. The addressee is a fixed 9-character field (padded with spaces if necessary) following the **:** Data Type Identifier. The addressee field is followed by another **:**, then the text of the message.

The message text may be up to 67 characters long, and may contain any printable ASCII characters except **|**, **~** or **{**.

A message may also have an optional message identifier, which is appended to the message text. The message identifier consists of the character **{** followed by a message number (up to 5 alphanumeric characters, no spaces) to identify the message.

Messages *without* a message identifier are not to be acknowledged.

Messages *with* a message identifier are intended to be acknowledged by the addressee. The sending station will repeatedly send the message until it receives an acknowledgement, or it is canceled, or it times out.

Message Format

	:	Addressee	:	Message Text (max 67 chars)	Message ID {	Message No xxxxx
Bytes:	1	9	1	0-67	1	1-5

Examples

`:WU2Z␣␣␣␣␣:Testing` A message for WU2Z, containing the text “Testing”, no acknowledgement expected. (Note the filler spaces in the 9-character addressee field).

`:WU2Z␣␣␣␣␣:Testing{003` The same message, Message No=003, acknowledgement expected.

`:EMAIL␣␣␣␣:msproul@ap.org Test email` An e-mail message (Note: This is an example of how such a message could be constructed. APRS itself does not support e-mail delivery)

Message Acknowledgement

A message acknowledgement is similar to a message, except that the message text field contains just the letters **ack**, and this is followed by the Message Number being acknowledged.

Message Acknowledgement Format				
:	***Addressee***	**:**	**ack**	***Message No*** xxxxx
Bytes: 1	9	1	3	1–5
Example :KB2ICI-14:ack003				

Message Rejection

If a station is unable to accept a message, it can send a **rej** message instead of an **ack** message:

Message Rejection Format				
:	***Addressee***	**:**	**rej**	***Message No*** xxxxx
Bytes: 1	9	1	3	1–5
Example :KB2ICI-14:rej003				

Multiple Acknowledgements

If a station receives a particular message more than once, it will respond with an acknowledgement for each instance of the message.

If a station receives a message over a long path, it may respond with a reasonable number of multiple copies of the acknowledgement, to improve the chances of the originating station receiving at least one of the copies.

In either of these two situations, multiple message acknowledgements should be separated by at least 30 seconds (this is because some network components such as digipeaters will suppress duplicated messages within a 30-second period).

Message Groups

An APRS receiving station can specify special Message Groups, containing lists of callsigns that the station will read messages from (in addition to messages addressed to itself). Such Message Groups are defined internally by the user at the receiving station, and are used to filter received message traffic.

The receiving station will read all messages with the Addressee field set to `ALL`, `QST` or `CQ`.

The receiving station will only acknowledge messages addressed to itself, and not any messages received which were addressed to any group callsign.

Note: The receiving station will acknowledge all messages addressed to itself, even if it is operating in an Alternate Net (see Chapter 4: APRS Data in the AX.25 Destination and Source Address Fields).

General Bulletins

General bulletins are messages where the addressee consists of the letters **BLN** followed by a single-*digit* bulletin identifier, followed by 5 filler spaces. General bulletins are generally transmitted a few times an hour for a few hours, and typically contain time sensitive information (such as weather status).

Bulletin text may be up to 67 characters long, and may contain any printable ASCII characters except **|** or **~**.

General Bulletin Format					
:	**BLN**	***Bulletin ID*** `n`	**␣␣␣␣␣**	**:**	***Bulletin Text (max 67 characters)***
Bytes: 1	3	1	5	1	0-67
Example `:BLN3␣␣␣␣␣:Snow expected in Tampa RSN`					

Announcements

Announcements are similar to general bulletins, except that the letters **BLN** are followed by a single upper-case *letter* announcement identifier. Announcements are transmitted much less frequently than bulletins (but perhaps for several days), and although possibly timely in nature they are usually not time critical.

Announcements are typically made for situations leading up to an event, in contrast to bulletins which are typically used within the event.

Users should be alerted on arrival of a new bulletin or announcement.

Announcement Format					
:	**BLN**	***Announcement Identifier*** `x`	**␣␣␣␣␣**	**:**	***Announcement Text (max 67 characters)***
Bytes: 1	3	1	5	1	0-67
Example `:BLNQ␣␣␣␣␣:Mt St Helen digi will be QRT this weekend`					

Group Bulletins

Bulletins may be sent to *bulletin groups*. A bulletin group address consists of the letters **BLN**, followed by a single-*digit* group bulletin identifier, followed in turn by the name of the group (up to 5 characters long, with filler spaces to pad the name to 5 characters).

Group Bulletin Format

	:	BLN	Group Bulletin ID n	Group Name	:	Group Bulletin Text (max 67 characters)
Bytes:	1	3	1	5	1	0-67

Example

`:BLN4WX___:Stand by your snowplows` Group bulletin number 4 to the `WX` group. (Note the filler spaces in the group name).

A receiving station can specify a list of bulletin groups of interest. The list is defined internally by the user at the receiving station. If a group is selected from the list, the station will only copy bulletins for that group, plus any general bulletins. If the list is empty, all bulletins are received and generate alerts.

National Weather Service Bulletins

Standard APRS message formats can be used for a variety of other applications. For example, in the United States, special formatted messages addressed to the generic callsign **NWS-**`xxxxx` are used to highlight map areas involved in weather warnings, using the following format:

National Weather Service Bulletin Format

	:	NWS-xxxxx	:	NWS Bulletin Text
Bytes:	1	9	1	n

Example

`:NWS-WARN_:092010z,THUNDER_STORM,AR_ASHLEY,{S9JbA`

(Note: The "message identifier" `{S9JbA` at the end is for reference only, as receiving stations do not acknowledge bulletins).

NTS Radiograms

APRS can be used to transport NTS radiograms. This uses the existing APRS message format for backwards compatibility, by adding a 3-character NTS format identifier **N**x**** at the start of the APRS Message Text, as follows:

N#number****precedence****handling****originator****check****place****time****date
NAaddress_line1****address_line2****address_line3****address_line4
NPphone number
N1line 1 of NTS message text
N2line 2 of NTS message text
N3line 3 of NTS message text
N4line 4 of NTS message text
N5line 5 of NTS message text
N6line 6 of NTS message text
NSSignature block
NRReceived from****date_time****sent_to****date_time

All of these fields comes from the ARRL NTS Radiogram form and are described in the NTS handbook.

Each message line is addressed to the same station.

The **N#**, **NA** and **NR** lines are multiple fields combined for APRS transmission efficiency. The backslash separator is used so that conventional forward slashes may be embedded in messages. (The backslash does not exist in the RTTY or CW alphabets, so it therefore cannot appear in an NTS radiogram).

Each line may be up 67 characters long, including the 3-character NTS format identifier. Lines in excess of 67 characters will be truncated.

There is a maximum of 6 lines of NTS message text.

Note: The **N#**, **NA**, **NS** and **NR** fields are required. The others are optional.

Serialization of each line is handled by the normal APRS Message ID **{**xxxxx.

An APRS application is not required to understand or generate these messages. The information can be read and understood in the normal message display.

Obsolete Bulletin and Announcement Format

Some stations transmit bulletins and announcements without the **:** APRS Data Type Identifier. This format is obsolete. Some software may still decode such data as a bulletin or announcement, but it should now be interpreted as a Status Report.

Bulletin and Announcement Implementation Recommendations

Bulletins and announcements are seen as a way for all participants in an event/emergency/net to see all common information posted to the group. In this sense they are visualized as a mountain-top billboard or a bulletin board on the wall of an Emergency Operations Control Center.

Information that everyone must see is to be posted there. Information is added and removed. Space is limited. Only so much information can be posted before it becomes too busy for anyone to see everything. Thus things are supposed to be posted, updated, and cleared to keep the big billboard uncluttered and current with what everyone needs to know at the present time. It should not be cluttered with obsolete information.

This can be implemented in an APRS display system as a "Bulletin Screen". Everyone has this screen, and anyone can post or update lines on this screen. At any instant, everyone in the network sees exactly the same screen. Everything is arranged and displayed in exactly the same way. Thus, everyone, everywhere is looking at the same mountain-top billboard or bulletin board. There is no ambiguity as to who sees what information, in what order at what time.

To make this work, a number of issues should be considered:

- **Sorting**: Bulletins/Announcements are almost always multi-line, and may arrive out of sequence. They must be sorted before presentation on the Bulletin Screen, and re-sorted if necessary when each new line arrives. This includes sorting by originating callsign and Bulletin/ Announcement ID.
- **Replacement**: Stations sending bulletins/announcements can send new lines to replace lines sent earlier, re-using the original Bulletin/ Announcement IDs. (Note: It is only necessary to re-send replacement lines. It is not necessary to re-send the whole bulletin/announcement). Receipt of a new line with the same Bulletin/Announcement ID as one already received from the same station should result in the existing line being overwritten (replaced).
- **Clearing**: A user should be able to clear any or all of the bulletins/ announcements from the Bulletin Screen once he has read them. Any bulletins/announcements that are still valid will re-appear in due course because of the way they are redundantly re-transmitted.
- **Alerts**: On receipt of any new or replacement line for the Bulletin Screen, an alarm should be sounded and re-sounded periodically until the user acknowledges it. Thus, this vital information is "pushed" to the operator. There is no excuse for not being aware of the current bulletin/ announcement state — this is important in the hurried and crisis-laden scenario of an APRS event.
- **Logging**: Old bulletins/announcements should be logged in sequential APRS log files in case they are subsequently needed.

15 STATION CAPABILITIES, QUERIES AND RESPONSES

Station Capabilities

A station may define a set of one or more attributes of the station, known as Station Capabilities. The station transmits its capabilities in response to an IGATE query (see below), using the **<** Data Type Identifier.

Each capability is a TOKEN or a TOKEN=VALUE pair. More than one capability may be on a line, with each capability separated by a comma.

Currently defined capabilities include:

```
IGATE,MSG_CNT=n,LOC_CNT=n
```

where `IGATE` defines the station as an IGate, `MSG_CNT` is the number of messages transmitted, and `LOC_CNT` is the number of "local" stations (those to which the IGate will pass messages in the local RF network).

Queries and Responses

There are two types of APRS queries. One is general to all stations and the other is in a message format directed to a single individual station.

Queries always begin with a **?**, are one-time transmissions, do not have a message identifier and should not be acknowledged. Similarly the responses to queries are one-time transmissions that also do not have a message identifier, so that they too are not acknowledged.

Each query contains a Query Type (in upper-case). The following Query Types and expected responses are supported:

Query Type	*Query*	*Response*
APRS	General — All stations query	Station's position and status
APRSD	Directed — Query an individual station for stations heard direct	List of stations heard direct
APRSH	Directed — Query if an individual station has heard a particular station	Position of heard station as an APRS Object, plus heard statistics for the last 8 hours
APRSM	Directed — Query an individual station for outstanding unacknowledged or undelivered messages	All outstanding messages for the querying station
APRSO	Directed — Query an individual station for its Objects	Station's Objects
APRSP	Directed — Query an individual station for its position	Station's position
APRSS	Directed — Query an individual station for its status	Station's status
APRST or **PING?**	Directed — Query an individual station for a trace (i.e. path by which the packet was heard)	Route trace
IGATE	General — Query all Internet Gateways	IGate station capabilities
WX	General — Query all weather stations	Weather report (and the station's position if it is not included in the Weather Report)

If a queried station has no relevant information to include in a response, it need not respond.

A queried station should ignore any query that it does not recognize.

General Queries

The format of a general query is as follows:

General Query Format							
			Target Footprint				
?	Query Type	?	Lat	,	Long	,	Radius
Bytes: 1	n	1	n	1	n	1	4

Examples

Query	Typical Response
`?APRS?` General query, with standard posit and status reply.	`/092345z4903.50N/07201.75W>` `>092345zNet Control Center`
`?APRS?␣34.02,-117.15,0200` General query for stations within a target footprint of radius 200 miles centered on 34.02 degrees north, 117.15 degrees west, with standard posit and status reply. (Note the leading space in the latitude, as its value is positive, see below).	`/3402.78N11714.02W-` `>Digi on low power`
`?IGATE?` General query for IGate stations, with a Station Capabilities reply.	`<IGATE,MSG_CNT=43,LOC_CNT=14`
`?WX?` Query for weather stations, with a standard Weather Report reply (without a position), followed by a standard posit.	`_10090556c220s004g005t077…` `/090556z4903.50N/07201.75W>`

In the case of an `?APRS?` query for stations within a particular target footprint, the latitude and longitude parameters are in *floating point* degrees (*not* in APRS lat/long position format).

- North and east coordinates are positive values, indicated by a leading ␣ (space).
- South and west coordinates are negative values.
- The radius of the footprint is in miles, expressed as a fixed 4-digit number in whole miles.

All stations inside the specified coverage circle should respond with a Position Report and a Status Report.

Directed Station Queries

Queries addressed to individual stations are in APRS message format (except that they never include a message identifier). The addressee is the callsign of the station being queried.

The message text is the Query Type. This is followed optionally by another callsign — this callsign does not need filler spaces as it is at the end of the data.

Directed Station Query Format

:	*Addressee*	:	?	*Query Type*	*Callsign of Heard Station*
Bytes: 1	9	1	1	5	0-9

Examples

Query	Typical Response
`:KH2Z␣␣␣␣␣:?APRSD` A query asking KH2Z what stations he has heard direct.	`:N8UR␣␣␣␣␣:Directs=␣WA1LOU␣WD5IVD…`
`:KH2Z␣␣␣␣␣:?APRSH␣N0QBF␣␣␣␣` A query asking for the number of times N0QBF was heard in each of the last 8 hours. (Note the trailing spaces in the callsign following `APRSH`, padding the callsign to 9 characters).	`:N8UR␣␣␣␣␣:N0QBF␣HEARD:␣1␣3␣2␣.␣.␣4␣5␣6`
`:KH2Z␣␣␣␣␣:?APRSM` A query asking KH2Z for any unacknowledged or undelivered messages for him. KH2Z responds with all such messages.	`:N8UR␣␣␣␣␣:Testing{003`
`:KH2Z␣␣␣␣␣:?APRSO` A query asking for KH2Z's APRS Objects.	`;LEADER␣␣␣*092345z4903.50N/07201.75W>`
`:KH2Z␣␣␣␣␣:?APRSP` A query asking for KH2Z's position.	`/092345z4903.50N/07201.75W>`
`:KH2Z␣␣␣␣␣:?APRSS` A query asking for KH2Z's status.	`>092345zNet Control Center`
`:KH2Z␣␣␣␣␣:?APRST` A query asking KH2Z for a trace of the route taken to reach him.	`:N8UR␣␣␣␣␣:KH2Z>APRS,DIGI1,WIDE*:`
`:KH2Z␣␣␣␣␣:?PING?` The same query, using PING instead of APRST.	`:N8UR␣␣␣␣␣:KH2Z>APRS,DIGI1,WIDE*:`

16 STATUS REPORTS

A Status Report announces the station's current mission or any other single line status to everyone. The report is contained in the AX.25 Information field, and starts with the **>** APRS Data Type Identifier.

The report may optionally contain a timestamp.

Note: The timestamp can *only* be in DHM *zulu* format.

The status text occupies the rest of the Information field, and may be up to 62 characters long (if there is no timestamp in the report) or 55 characters (if there is a timestamp). The text may contain any printable ASCII characters except **|** or **~**.

	Status Report Format		
	>	***Time DHM z***	***Status Text (max 62 chars if no timestamp, or 55 chars if there is a timestamp)***
Bytes:	1	7	0-62 or 0-55

Examples

```
>Net Control Center            without timestamp.
>092345zNet Control Center     with timestamp.
```

Although the status will usually be plain language text, there are two cases where the report can contain special information which can be decoded:

- Beam Heading and Power
- Maidenhead grid locator

Status Report with Beam Heading and Effective Radiated Power

It is useful to include beam heading and ERP in packets in meteor scatter work. To keep packets as short as possible, these parameters are encoded into two characters, as follows:

H = beam heading / 10
(H=0–9 for 0–90 degrees, and A–Z for 100–350 degrees).

P = ERP code.

ERP	P
10w	1
40	2
90	3
160	4
250	5
360	6
490	7
640	8
810	9

ERP	P
1000w	:
1210	;
1440	<
1690	=
1960	>
2250	?
2560	@
2890	A
3240	B

ERP	P
3610w	C
4000	D
4410	E
4840	F
5290	G
5760	H
6250	I
6760	J
7290	K

The HP value appears as the *last* two characters of the status text, preceded by the **^** character — for example, **^B7** means a beam heading of 110 degrees and an ERP of 490 watts.

The HP value may be combined with the Maidenhead grid locator (as described below), or with any other plain language status text.

Status Report with Maidenhead Grid Locator

The Maidenhead grid locator may be 4 or 6 characters long, and must immediately follow the **>** Data Type Identifier.

All letters must be transmitted in upper case. Letters may be received in upper case or lower case.

The Symbol Table Identifier and Symbol Code follow the locator.

If the report also contains status text, the first character of the text *must* be a space.

A Status Report with Maidenhead locator can not have a timestamp.

Status Report Format — with Maidenhead Grid Locator							
		Maidenhead Locator			*Sym Table ID*	*Symbol Code*	*Status Text (starting with a space) (max 54 chars)*
	>	GG	nn	gg			␣
Bytes:	1	2	2	2	1	1	1-54

Examples

```
>IO91SX/G
>IO91/G
>IO91SX/-␣My house        (Note the space ␣ at the start of the status text).
>IO91SX/-␣^B7             Meteor Scatter beam heading = 110 degrees, ERP = 490 watts.
```

Transmitting Status Reports

Each station should only transmit a Status Report once every net cycle time (i.e. once every 10, 20 or 30 minutes), or in response to a query.

17 NETWORK TUNNELING AND THIRD-PARTY DIGIPEATING

Third-Party Networks

APRS provides a mechanism for formatting packets that are to be transported through third-party (i.e. non AX.25) networks, such as the Internet, an Ethernet LAN or a direct wire connection.

These networks do not understand APRS source, destination and digipeater addresses, so it is necessary to send them as data, along with the original data being transmitted.

Source Path Header

Prior to sending an APRS packet into the third-party network, the APRS address path is prepended to the Data Type Identifier and the rest of the original data.

The prepended address path is known as the Source Path Header. It consists of the source, destination and digipeater callsigns, with associated SSIDs.

The main purpose of introducing the Source Path Header is to allow receiving stations on the far side of the third-party network to identify the sender — this is needed when acknowledging receipt of a message, for example. Knowledge of the source path is also useful in diagnosing network problems.

Data with Source Path Header		
Source Path Header	***Data Type ID***	***Rest of the original data***
Bytes: n	1	n

The Source Path Header may be in either of two formats, known as the "TNC-2" format and the "AEA" format (so called because when TNC-2 or AEA-compatible TNCs are operating in terminal MONitor mode they automatically produce headers in these formats).

The APRS Working Group has agreed to move towards standardization on the "TNC-2" format in future implementations.

In most cases, AEA TNCs will produce Source Path Headers in "TNC-2" format when BBSMSGS is set to ON.

The formats of these headers are as follows:

Source Path Header — "TNC-2" Format
An asterisk follows the digipeater callsign heard.

	Source Callsign (-SSID)	>	***Destination Callsign*** (-SSID)	***0-8 Digipeaters*** , ***Digipeater Callsign*** (-SSID)(*)	:
Bytes:	1-9	1	1-9	0-81	1

Example

```
WB4APR-14>APRS,RELAY*,WIDE:
```
(WIDE digipeater "unused")

Source Path Header — "AEA" Format
An asterisk follows the source or digipeater callsign heard.

	Source Callsign (-SSID)(*)	***0-8 Digipeaters*** > ***Digipeater Callsign*** (-SSID)(*)	>	***Destination Callsign*** (-SSID)	:
Bytes:	1-10	0-81	1	1-9	1

Example

```
WB4APR-14>RELAY*>WIDE>APRS:
```
(WIDE digipeater "unused")

In both formats, the SSID may be omitted if it is –0.

In both formats, the callsign of the digipeater from which the incoming packet was heard is indicated with an asterisk. (Alternatively, for "AEA" format only, the asterisk will follow the source callsign if the packet was heard direct from there).

Any digipeaters following the callsign of the station from which the packet was heard are termed "unused". These unused digipeaters are stripped out when building a Third-Party Header (see below).

Third-Party Header

After a packet emerges from a third-party network, the receiving gateway station modifies it (by inserting a **}** Third-Party Data Type Identifier and modifying the Source Path Header) before transmitting it on the local APRS network.

The modified Source Path Header is called the Third-Party Header.

	Third-party Format		
	}	***Third-Party Header***	***Rest of the original data***
Bytes:	1	n	n

In a similar way to the Source Path Header, The Third-Party Header can be in either of two formats: "TNC-2" or "AEA" format.

	Third Party Header — "TNC-2" format						
	Source Path Header (without "unused" digipeaters, * or :)	**,**	***Third-Party Network Identifier ("callsign")***	**,**	***Callsign of Receiving Gateway Station (-SSID)***	*****	**:**
Bytes:	3-99	1	1-9	1	1-9	1	1
	Example `WB4APR-14>APRS,RELAY,TCPIP,G9RXG*:`						

	Third Party Header — "AEA" format							
	Source Path Header (without "unused" digipeaters, destination, * or :)	***Third-Party Network Identifier ("callsign")***	**>**	***Callsign of Receiving Gateway Station (-SSID)***	*****	**>**	***Destination Callsign from Source Path Header (-SSID)***	**:**
Bytes:	2-90	1-9	1	1-9	1	1	1-9	1
	Example `WB4APR-14>RELAY>TCPIP>G9RXG*>APRS:`							

In both cases, the "unused" digipeater callsigns (i.e. those digipeater callsigns after the asterisk) in the original Source Path Header are stripped out. The asterisk itself is also stripped out of the Source Path Header.

Then two additional callsigns are inserted:

- The Third-Party Network Identifier (e.g. `TCPIP`). This is a dummy "callsign" that identifies the nature of the third-party network.
- The callsign of the receiving gateway station, followed by an asterisk.

Action on Receiving a Third-Party packet

When another station receives a third-party packet, it can extract the callsign of the original sending station from the Third-Party Header, if it is needed to acknowledge receipt of a message.

The other addresses in the Third-Party Header may be useful for network diagnostic purposes.

An Example of Sending a Message through the Internet

The Scenario:

- WB4APR-14 wants to send a message via the Internet to G3NRW.
- The nearest Internet gateway to WB4APR-14 is K4HG, reachable via a `RELAY,WIDE` path.
- The nearest Internet gateway to G3NRW is G9RXG.

The Process:

- In the normal way, WB4APR-14 builds a message packet that contains:
  `:G3NRW____:Hi Ian{001`
- WB4APR-14 transmits the packet via his UNPROTO path `RELAY,WIDE`.
- The Internet gateway K4HG happens to receive this packet from the `RELAY` digipeater in the path.
- K4HG builds a new packet that contains the source path and the original message:
  `WB4APR-14>APRS,RELAY*,WIDE::G3NRW____:Hi Ian{001`
- K4HG sends this packet (using telnet) to an APRServer on the Internet.
- All Internet gateways throughout the world that are connected to the APRServe network (including G9RXG) receive the packet.
- G9RXG converts the packet into a Third-Party packet:
  `}WB4APR-14>APRS,RELAY,TCPIP,G9RXG*::G3NRW____:Hi Ian{001`

  Note that the `WIDE` digipeater was stripped out of the header because it was unused.
- G9RXG transmits the packet over the local APRS network.
- G3NRW receives the packet, strips out the Third-Party Header, and discovers that the packet contains a message for him. From the header, G3NRW then establishes that the acknowledgement is to go back to WB4APR-14.

18 USER-DEFINED DATA FORMAT

The APRS protocol defines many different data formats, but it cannot anticipate every possible data type that programmers may wish to send. The User-Defined data format is designed to fill these gaps. Under this system, program authors are free to send data in any format they choose.

The data in the AX.25 Information field consists of a three-character header:

{	APRS Data Type Identifier.
U	A one-character User ID.
X	A one-character user-defined packet type.

The APRS Working Group will issue User IDs to program authors who express a need.

[Keep in mind there is a limited number of available User IDs, so please do not request one unless you have a true need. The Working Group may require an explanation of your need prior to issuing a character. If only one or two data formats are needed, those may be issued from a User ID pool].

For experimentation, or prior to being issued a User ID, anyone may utilize the User ID character of { without prior notification or approval (i.e. packets beginning with {{ are experimental, and may be sent by anyone).

Important Note: Although there is no restriction on the nature of user-defined data, it is highly recommended that it is represented in printable 7-bit ASCII character form.

User-Defined Data Format

	{	*User ID* U	*User-Defined Packet Type* X	*User-defined data (printable ASCII recommended)*
Bytes:	1	1	1	n

Examples

{Q1qwerty	User ID = Q, User-defined packet type = 1.
{{zasdfg	User ID undefined (experimental), User-defined packet type = z.

This is envisioned as a way for authors to experiment and build in features specific to their programs, without the danger of a non-standard packet crashing other authors' programs. In keeping with the spirit of the APRS protocol, authors are encouraged to make these formats public. The APRS Working Group will maintain a web site defining all of the assigned User IDs, and either the packet formats provided by the author, or links to their

own web sites which define their formats.

Generally, all formats using this method will be considered optional. No program is required to decode any of these packets, and must ignore any it does not decode. However, it is possible that in the future some of these formats may prove to be of sufficient utility and interest to the entire APRS community that they will be specifically included in future versions of the APRS protocol.

19 OTHER PACKETS

Invalid Data or Test Data Packets

To indicate that a packet contains invalid data, or test data that does not conform to any standard APRS format, the , Data Type Identifier is used.

For example, the Mic-E unit will generate such a packet if it detects that a received GPS sentence is not valid.

Invalid Data / Test Data Format	
,	*Invalid Data or Test Data*
Bytes: 1	n

Example

```
,191146,V,4214.2466,N,07303.5181,W,417.238,114.5,091099,14.7,W/GPS FIX
```

Invalid GPS data from a Mic-E unit. The unit has interpreted the **V** character in the received sentence to mean the data is invalid, and has stripped out the `$GPRMC` header.

All Other Packets

Packets that do not meet any of the formats described in this document are assumed to be non-APRS beacons. Programs can decide to handle these, or ignore them, but they must be able to process them without ill effects.

APRS programs may treat such packets as APRS Status Reports. This allows APRS to accept any UI packet addressed to the typical beacon address to be captured as a status message. Typical TNC ID packets fall into this category. Once a proper Status Report (with the APRS Data Type Identifier **>**) has been received from a station it will not be overwritten by other non-APRS packets from that station.

20 APRS SYMBOLS

Three Methods

There are three methods of specifying an APRS symbol (display icon):

- In the AX.25 Information field.
- In the AX.25 Destination Address.
- In the SSID of the AX.25 Source Address.

The preferred method is to include the symbol in the Information field. However, where this is not possible (for example, in stand-alone trackers with no means of introducing the symbol into the Information field), either of the other two methods may be used instead.

The Symbol Tables

There are two APRS Symbol Tables:

- Primary Symbol Table
- Alternate Symbol Table

See Appendix 2 for a full listing of these tables.

The essential difference between the Primary and Alternate Symbol Tables is that some of the symbols in the Alternate Symbol Table can be overlaid with an alphanumeric character. For example, a “car” icon in the Alternate Symbol Table could be overlaid with the digit “3”, to indicate it is car #3.

Symbols capable of taking an overlay are marked as ***[with overlay]***.

None of the symbols in the Primary Symbol Table can be overlaid.

In the tables, each symbol is coded in three ways:

- **/$** or **\$** — for symbols in the Information field.
- **GPSxyz** — for generic Destination addresses containing symbols.
- **GPSCnn** or **GPSEnn** — another form of generic Destination addresses containing systems.

In addition, 15 of the symbols in the Primary Symbol Table have an associated SSID (e.g. a small aircraft has SSID -7). The SSID is intended for use in the AX.25 Source Address of stand-alone trackers which have no other means of specifying the symbol.

Symbols in the AX.25 Information Field

A symbol in the AX.25 Information field is a combination of a one-character Symbol Table Identifier and a one-character Symbol Code.

For example, in the Position Report:

```
@092345z4903.50N/07201.75W>088/036…
```

the forward slash **/** is the Symbol Table Identifier and the **>** character is the Symbol Code (in this case representing a "car" icon) from the selected table.

The Symbol Table Identifier character selects one of the two Symbol Tables, or it may be used as single-character (alpha or numeric) overlay, as follows:

Symbol Table Identifier	*Selected Table or Overlay Symbol*
/	Primary Symbol Table (mostly stations)
****	Alternate Symbol Table (mostly Objects)
0-9	Numeric overlay. Symbol from Alternate Symbol Table (*uncompressed* lat/long data format)
a-j	Numeric overlay. Symbol from Alternate Symbol Table (*compressed* lat/long data format only). i.e. a-j maps to 0-9
A-Z	Alpha overlay. Symbol from Alternate Symbol Table

In the generic case, a symbol from the Primary Symbol Table is represented as the character-pair **/$**, and a symbol from the Alternate Symbol Table as **\$**.

Overlays with Symbols in the AX.25 Information Field

Where the Symbol Table Identifier is `0-9` or `A-Z` (or `a-j` with *compressed* position data only), the symbol comes from the *Alternate* Symbol Table, and is overlaid with the identifier (as a single digit or a capital letter).

For example, in the *uncompressed* Position Report:

```
@092345z4903.50N307201.75W>…
```

the digit **3** following the latitude will cause the number "3" to be overlaid on top of the "car" icon (**Note**: Because the symbol is overlaid, the **>** Symbol Code here comes from the *Alternate* Symbol Table).

Similarly, to overlay a "car" icon with the letter "B" in a *compressed* Position Report, the report will look something like:

```
=BL!!<*e7 >7P[
```

However, in a *compressed* Position Report, it is not permissible to use a *numeric* Symbol Table Identifier (0-9) — *compressed* positions never start with a digit. If a numeric overlay is required, the report must use a lower-case letter instead (in the range **a-j**) as the Symbol Table Identifier. The lower-case letter is then mapped to the digits **0-9** (i.e. a=0, b=1, c=2, d=3 etc).

Thus, in the *compressed* Position Report:

```
=d5L!!<*e7 >7P[
```

the letter **d** maps to overlay character "3".

As noted above, not all symbols from the Alternate Symbol Table may be overlaid in this way — those that can be overlaid are marked as ***[with overlay]*** in Appendix 2. This means that they are *capable* of taking an overlay, but they do not necessarily need to have one. Thus, for example, the following report uses the car symbol from the Alternate Symbol Table, but does not display an overlay:

```
@092345z4903.50N\07201.75W>…
```

Symbols in the AX.25 Destination Address

Where it is not possible to include a symbol in the Information field, the symbol may be specified in the AX.25 Destination Address instead, using the following generic destination addresses: `GPSxyz`, `GPSCnn`, `GPSEnn`, `SPCxyz` and `SYMxyz`.

The characters `xy` and `nn` refer to entries in the APRS Symbol Tables. For example, from the Primary Symbol Table, a tracker could use the Destination Address `GPS`**`MV␣`** or `GPS`**`30`** to specify a "car" icon.

The character `z` specifies the overlay character (where permitted), or is a **␣** (space) — the space is a filler character, as all AX.25 addresses must be exactly 6 characters long.

The `GPS/SPC/SYMxy␣` and `GPSCnn/GPSEnn` addresses can be used interchangeably. Thus, for example, `GPSBM␣` , `SPCBM␣` , `SYMBM␣` and `GPSC12` all specify a "Boy Scouts" icon (from the Primary Symbol Table), and `GPSOM␣` , `SPCOM␣` , `SYMOM␣` and `GPSE12` all specify a "Girl Scouts" icon (from the Alternate Symbol Table).

Overlays with Symbols in the AX.25 Destination Address

If the `z` character in a `GPSxyz`, `SPCxyz` or `SYMxyz` address is not a space, it specifies an alphanumeric overlay character, in the range `0-9` or `A-Z`.

Overlays can only be used with symbols from the Alternate Symbol Table marked with the legend ***[with overlay]***.

For example, if the "car" icon is to be overlaid with a digit "3", the Destination Address will be `GPS`**`NV3`**.

However, even if the address is overlay-capable, it is not actually necessary to specify an overlay; e.g. `GPS`**`NV␣`**.

`GPSCnn` and `GPSEnn` symbols can not have overlays.

Symbol in the Source Address SSID

Where it is not possible to include a symbol in the Information field or in the Destination Address, the symbol may be specified in the SSID of the Source Address instead:

SSID-Specified Icons in the AX.25 Source Address Field

SSID	Icon
-0	*[no icon]*
-1	Ambulance
-2	Bus
-3	Fire Truck
-4	Bicycle
-5	Yacht
-6	Helicopter
-7	Small Aircraft

SSID	Icon
-8	Ship (power boat)
-9	Car
-10	Motorcycle
-11	Balloon
-12	Jeep
-13	Recreational Vehicle
-14	Truck
-15	Van

Symbol Precedence

APRS packets should not contain more than one symbol. However, it is conceivably possible to (erroneously) construct a packet containing up to three different symbols.

For example:

	Source Address SSID	Destination Address	Information Field
	G3NRW**-7**	GPS**MV**	!0123.45N**/**01234.56W**j**
Symbol	Small Aircraft	Car	Jeep

In such a situation:

- The symbol in the Information field takes precedence over any other symbol.
- If there is no symbol in the Information field, the symbol in the Destination Address takes precedence over the symbol in the Source Address SSID.

APPENDIX 1: APRS DATA FORMATS

This Appendix contains format diagrams for all APRS data formats. The gray fields are optional. Shaded (yellow) characters are literal ASCII characters.

	AX.25 UI-FRAME FORMAT								
	Flag	Destination Address	Source Address	Digipeater Addresses (0-8)	Control Field (UI)	Protocol ID	INFORMATION FIELD	FCS	Flag
Bytes:	1	7	7	0–56	1	1	1–256	2	2

	Generic APRS Information Field			
	Data Type ID	APRS Data	APRS Data Extension	Comment
Bytes:	1	n	7	n

	Lat/Long Position Report Format — without Timestamp					
	! or =	Lat	Sym Table ID	Long	Symbol Code	Comment (max 43 chars)
Bytes:	1	8	1	9	1	0-43

	Lat/Long Position Report Format — with Timestamp						
	/ or @	Time DHM / HMS	Lat	Sym Table ID	Long	Symbol Code	Comment (max 43 chars)
Bytes:	1	7	8	1	9	1	0-43

	Lat/Long Position Report Format — with Data Extension (no Timestamp)						
	! or =	Lat	Sym Table ID	Long	Symbol Code	Course/Speed Power/Height/Gain/Dir Radio Range DF Signal Strength	Comment (max 36 chars)
Bytes:	1	8	1	9	1	7	0-36

Lat/Long Position Report Format — with Data Extension and Timestamp								
	/ or *@*	*Time DHM / HMS*	*Lat*	*Sym Table ID*	*Long*	*Symbol Code*	*Course/Speed* *Power/Height/Gain/Dir* *Radio Range* *DF Signal Strength*	*Comment (max 36 chars)*
Bytes:	1	7	8	1	9	1	7	0-36

Maidenhead Locator Beacon				
	[	*Grid Locator*	*]*	*Comment*
Bytes:	1	4 or 6	1	n

Raw NMEA Position Report Format		
	NMEA Received Sentence	
	$	…, …, …, …, …, …, …, …, …, …
Bytes:	1	25-209

DF Report Format — without Timestamp								
	! or *=*	*Lat*	*Sym Table ID* */*	*Long*	*Symbol Code* **	*Course/Speed* *Power/Height/Gain/Dir* *Radio Range* *DF Signal Strength*	*/BRG/NRQ*	*Comment (max 28 chars)*
Bytes:	1	8	1	9	1	7	8	0-28

DF Report Format — with Timestamp									
	/ or *@*	*Time DHM / HMS*	*Lat*	*Sym Table ID* */*	*Long*	*Symbol Code* **	*Course/Speed* *Power/Height/Gain/Dir* *Radio Range* *DF Signal Strength*	*/BRG/NRQ*	*Comment (max 28 chars)*
Bytes:	1	7	8	1	9	1	7	8	0-28

Compressed Lat/Long Position Report Format — no Timestamp

	! or =	Sym Table ID	Comp Lat YYYY	Comp Long XXXX	Symbol Code	Compressed Course/Speed / Compressed Radio Range / Compressed Altitude	Comp Type T	Comment (max 40 chars)
Bytes:	1	1	4	4	1	2	1	0-40

Compressed Lat/Long Position Report Format — with Timestamp

	/ or @	Time DHM / HMS	Sym Table ID	Comp Lat YYYY	Comp Long XXXX	Symbol Code	Compressed Course/Speed / Compressed Radio Range / Compressed Altitude	Comp Type T	Comment (max 40 chars)
Bytes:	1	7	1	4	4	1	2	1	0-40

Compression Type (T) Byte Format

Bit:	7	6	5	4	3	2	1	0
	Not used	*Not used*	***GPS Fix***	***NMEA Source***		***Compression Origin***		
Value:	0	0	0 = old (last) 1 = current	0 0 = other 0 1 = GLL 1 0 = GGA 1 1 = RMC		0 0 0 = Compressed 0 0 1 = TNC BText 0 1 0 = Software (DOS/Mac/Win/+SA) 0 1 1 = [tbd] 1 0 0 = KPC3 1 0 1 = Pico 1 1 0 = Other tracker [tbd] 1 1 1 = Digipeater conversion		

Mic-E Data — DESTINATION ADDRESS FIELD Format

	Lat Digit 1 + Message Bit A	Lat Digit 2 + Message Bit B	Lat Digit 3 + Message Bit C	Lat Digit 4 + N/S Lat Indicator	Lat Digit 5 + Longitude Offset	Lat Digit 6 + W/E Long Indicator	APRS Digi Path Code
Bytes:	1	1	1	1	1	1	1

Mic-E Data — INFORMATION FIELD Format

	Data Type ID	Longitude			Speed and Course			Symbol Code	Sym Table ID	Mic-E Telemetry Data / Mic-E Status Text
		d+28	m+28	h+28	SP+28	DC+28	SE+28			
Bytes:	1	1	1	1	1	1	1	1	1	n

Object Report Format — with Lat/Long position										
	;	***Object Name***	***** or ***_***	***Time DHM / HMS***	***Lat***	***Sym Table ID***	***Long***	***Symbol Code***	***Course/Speed*** ***Power/Height/Gain/Dir*** ***Radio Range*** ***DF Signal Strength*** ***Area Object***	***Comment (max 36 chars with Data Extension, or 43 without)***
Bytes:	1	9	1	7	8	1	9	1	7	0-36/43

Object Report Format — with Compressed Lat/Long position						
	;	***Object Name***	***** or ***_***	***Time DHM / HMS***	***Compressed Position Data*** `/YYYYXXXX$csT`	***Comment***
Bytes:	1	9	1	7	13	43

Item Report Format — with Lat/Long position									
	)	***Item Name***	***!*** or ***_***	***Lat***	***Sym Table ID***	***Long***	***Symbol Code***	***Course/Speed*** ***Power/Height/Gain/Dir*** ***Radio Range*** ***DF Signal Strength*** ***Area Object***	***Comment (max 36 chars with Data Extension, or 43 without)***
Bytes:	1	3-9	1	8	1	9	1	7	0-36/43

Item Report Format — with Compressed Lat/Long position						
	)	***Item Name***	***!*** or ***_***	***Compressed Position Data*** `/YYYYXXXX$csT`	***Comment***	
Bytes:	1	3-9	1	13	43	

Raw Weather Report Format		
	! *or* ***#*** *or* ***$*** *or* *****	***Raw Weather Data***
Bytes:	1	n

Positionless Weather Report Format				
_	Time MDHM	Positionless Weather Data	APRS Software S	WX Unit uuuu
Bytes: 1	8	n	1	2-4

Positionless Weather Data								
Wind Direction cccc	Wind Speed ssss	Gust gggg	Temp ttt	Rain Last Hr rrr	Rain Last 24 Hrs ppp	Rain Since Midnight PPP	Humidity hh	Barometric Pressure bbbbb
Bytes: 4	4	4	4	4	4	4	3	5

Complete Weather Report Format — with Lat/Long position, no Timestamp								
! or =	Lat	Sym Table ID	Long	Symbol Code _	Wind Directn/ Speed	Weather Data	APRS Software S	WX Unit uuuu
Bytes: 1	8	1	9	1	7	n	1	2-4

Complete Weather Report Format — with Lat/Long position and Timestamp									
/ or @	Time DHM / HMS	Lat	Sym Table ID	Long	Symbol Code _	Wind Directn/ Speed	Weather Data	APRS Software S	WX Unit uuuu
Bytes: 1	7	8	1	9	1	7	n	1	2-4

Complete Weather Report Format — with Compressed Lat/Long position, no Timestamp									
! or =	Sym Table ID	Comp Lat YYYY	Comp Long XXXX	Symbol Code _	Comp Wind Directn/ Speed	Comp Type T	Weather Data	APRS Software S	WX Unit uuuu
Bytes: 1	1	4	4	1	2	1	n	1	2-4

	Complete Weather Report Format — with Compressed Lat/Long position, with Timestamp										
	/ *or* **@**	***Time DHM / HMS***	***Sym Table ID***	***Comp Lat*** YYYY	***Comp Long*** XXXX	***Symbol Code*** _	***Comp Wind Directn/ Speed***	***Comp Type*** T	***Weather Data***	***APRS Software*** S	***WX Unit*** uuuu
Bytes:	1	7	1	4	4	1	2	1	n	1	2-4

	Complete Weather Report Format — with Object and Lat/Long position											
	*****	***Object Name***	*****	***Time DHM / HMS***	***Lat***	***Sym Table ID***	***Long***	***Symbol Code*** _	***Wind Directn/ Speed***	***Weather Data***	***APRS Software*** S	***WX Unit*** uuuu
Bytes:	1	9	1	7	8	1	9	1	7	n	1	2-4

	Storm Data									
	Direction	**/**	***Speed***	***Storm Type*** /ST	***Sustained Wind Speed*** /www	***Peak Wind Gusts*** ^GGG	***Central Pressure*** /pppp	***Radius Hurricane Winds*** >RRR	***Radius Tropical Storm Winds*** &rrr	***Radius Whole Gale*** %ggg
Bytes:	3	1	3	3	4	4	5	4	4	4

	Telemetry Report Format								
	T	***Sequence No*** #nnn,	***Analog Value 1*** aaa,	***Analog Value 2*** aaa,	***Analog Value 3*** aaa,	***Analog Value 4*** aaa,	***Analog Value 5*** aaa,	***Digital Value*** bbbbbbbb	***Comment***
Bytes:	1	5	4	4	4	4	4	8	n

	Telemetry Parameter Name Message Data Note the different byte counts, which include commas where shown. The list may stop at any field.													
	PARM.	***A1*** N...	***A2*** ,N...	***A3*** ,N...	***A4*** ,N...	***A5*** ,N...	***B1*** ,N...	***B2*** ,N...	***B3*** ,N...	***B4*** ,N...	***B5*** ,N...	***B6*** ,N...	***B7*** ,N...	***B8*** ,N...
Bytes:	5	1-7	1-7	1-6	1-6	1-5	1-6	1-5	1-4	1-4	1-4	1-3	1-3	1-3

Telemetry Unit/Label Message Data
Note the different byte counts, which include commas where shown. The list may stop at any field.

	UNIT.	***A1*** U...	***A2*** ,U...	***A3*** ,U...	***A4*** ,U...	***A5*** ,U...	***B1*** ,L...	***B2*** ,L...	***B3*** ,L...	***B4*** ,L...	***B5*** ,L...	***B6*** ,L...	***B7*** ,L...	***B8*** ,L...
Bytes:	5	1-7	1-7	1-6	1-6	1-5	1-6	1-5	1-4	1-4	1-4	1-3	1-3	1-3

Telemetry Equation Coefficients Message Data
The list may stop at any field. Value = $a x v^2 + b x v + c$

		A1			***A2***			***A3***			***A4***			***A5***		
	EQNS.	a	,b	,c	,a	,b	,c	,a	,b	,c	,a	,b	,c	,a	,b	,c
Bytes:	5	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

Telemetry Bit Sense/Project Name Message Data

	BITS.	***B1*** x	***B2*** x	***B3*** x	***B4*** x	***B5*** x	***B6*** x	***B7*** x	***B8*** x	***Project Title***
Bytes:	5	1	1	1	1	1	1	1	1	0-23

Message Format

	:	***Addressee***	**:**	***Message Text (max 67 chars)***	***Message ID***	
					{	***Message No*** xxxxx
Bytes:	1	9	1	0-67	1	1-5

Message Acknowledgement Format

	:	***Addressee***	**:**	**ack**	***Message No*** xxxxx
Bytes:	1	9	1	3	1–5

Message Rejection Format

	:	***Addressee***	**:**	**rej**	***Message No*** xxxxx
Bytes:	1	9	1	3	1–5

General Bulletin Format					
:	BLN	*Bulletin ID* n	_____	:	*Bulletin Text (max 67 characters)*
Bytes: 1	3	1	5	1	0-67

Announcement Format					
:	BLN	*Announcement Identifier* x	_____	:	*Announcement Text (max 67 characters)*
Bytes: 1	3	1	5	1	0-67

Group Bulletin Format					
:	BLN	*Group Bulletin ID* n	*Group Name*	:	*Group Bulletin Text (max 67 characters)*
Bytes: 1	3	1	5	1	0-67

National Weather Service Bulletin Format			
:	NWS-xxxxx	:	*NWS Bulletin Text*
Bytes: 1	9	1	n

General Query Format							
			Target Footprint				
?	*Query Type*	?	*Lat*	,	*Long*	,	*Radius*
Bytes: 1	n	1	n	1	n	1	4

Directed Station Query Format					
:	*Addressee*	:	?	*Query Type*	*Callsign of Heard Station*
Bytes: 1	9	1	1	5	0-9

	Status Report Format		
	>	Time DHM z	Status Text (max 62 chars if no timestamp, or 55 chars if there is a timestamp)
Bytes:	1	7	0-62 or 0-55

	Status Report Format — with Maidenhead Grid Locator						
		Maidenhead Locator			Sym Table ID	Symbol Code	
	>	GG	nn	gg			_ Status Text (starting with a space) (max 54 chars)
Bytes:	1	2	2	2	1	1	1-54

	Data with Source Path Header		
	Source Path Header	Data Type ID	Rest of the original data
Bytes:	n	1	n

Source Path Header — "TNC-2" Format An asterisk follows the digipeater callsign heard.					
			0-8 Digipeaters		
Source Callsign (-SSID)	>	Destination Callsign (-SSID)	,	Digipeater Callsign (-SSID)(*)	:
Bytes: 1-9	1	1-9	0-80		1

Source Path Header — "AEA" Format An asterisk follows the source or digipeater callsign heard.					
	0-8 Digipeaters				
Source Callsign (-SSID)(*)	>	Digipeater Callsign (-SSID)(*)	>	Destination Callsign (-SSID)	:
Bytes: 1-10	0-80		1	1-9	1

Third-party Format		
}	***Third-Party Header***	***Rest of the original data***
Bytes: 1	n	n

Third Party Header — "TNC-2" format						
Source Path Header (without "unused" digipeaters, * or :)	**,**	***Third-Party Network Identifier ("callsign")***	**,**	***Callsign of Receiving Gateway Station (-SSID)***	*****	**:**
Bytes: n	1	1-9	1	1-9	1	1

Third Party Header — "AEA" format							
Source Path Header (without "unused" digipeaters, destination, * or :)	***Third-Party Network Identifier ("callsign")***	**>**	***Callsign of Receiving Gateway Station (-SSID)***	*****	**>**	***Destination Callsign from Source Path Header (-SSID)***	**:**
Bytes: 2-90	1-9	1	1-9	1	1	1-9	1

User-Defined Data Format			
{	***User ID*** U	***User-Defined Packet Type*** X	***User-defined data (printable ASCII recommended)***
Bytes: 1	1	1	n

Invalid Data / Test Data Format	
,	***Invalid Data or Test Data***
Bytes: 1	n

Agrelo Format			
%	***Bearing*** nnn	**/**	***Quality*** n
Bytes: 1	3	1	1

APPENDIX 2: THE APRS SYMBOL TABLES

(Each highlighted character in the Alternate Symbol Table may be replaced with an overlay character).

PRIMARY SYMBOL TABLE				
/$	GPS xyz	GPS Cnn	Icon	
/!	BB␣	01	Police, Sheriff	
/"	BC␣	02	*[reserved]*	
/#	BD␣	03	Digi (green star with white center)	
/$	BE␣	04	Phone	
/%	BF␣	05	DX Cluster	
/&	BG␣	06	HF Gateway	
/'	BH␣	07	Small Aircraft (SSID –7)	
/(	BI␣	08	Mobile Satellite Groundstation	
/)	BJ␣	09		
/*	BK␣	10	Snowmobile	
/+	BL␣	11	Red Cross	
/,	BM␣	12	Boy Scouts	
/-	BN␣	13	House QTH (VHF)	
/.	BO␣	14	X	
//	BP␣	15	Dot	
/0	P0␣	16	Numerical Circle ⓪	Obsolete. Use the "Circle with overlay" symbol instead (code \0).
/1	P1␣	17	Numerical Circle ①	
/2	P2␣	18	Numerical Circle ②	
/3	P3␣	19	Numerical Circle ③	
/4	P4␣	20	Numerical Circle ④	
/5	P5␣	21	Numerical Circle ⑤	
/6	P6␣	22	Numerical Circle ⑥	
/7	P7␣	23	Numerical Circle ⑦	
/8	P8␣	24	Numerical Circle ⑧	
/9	P9␣	25	Numerical Circle ⑨	
/:	MR␣	26	Fire	
/;	MS␣	27	Campground	
/<	MT␣	28	Motorcycle (SSID –10)	
/=	MU␣	29	Railroad Engine	
/>	MV␣	30	Car (SSID –9)	
/?	MW␣	31	File Server	
/@	MX␣	32	Hurricane Future Prediction (dot)	
/A	PA␣	33	Aid Station	
/B	PB␣	34	BBS	
/C	PC␣	35	Canoe	

ALTERNATE SYMBOL TABLE			
\$	GPS xyz	GPS Enn	Icon
\!	OB␣	01	Emergency
\"	OC␣	02	*[reserved]*
\#	ODz	03	Digi (green star) *[with overlay]*
\$	OE␣	04	Bank or ATM (green box)
\%	OF␣	05	
\&	OGz	06	HF Gateway (diamond) *[w/ overlay]*
\'	OH␣	07	Crash Site
\(	OI␣	08	Cloudy
\)	OJ␣	09	
*	OK␣	10	Snow
\+	OL␣	11	Church
\,	OM␣	12	Girl Scouts
\-	ON␣	13	House (HF)
\.	OO␣	14	Unknown/indeterminate position
\/	OP␣	15	
\0	A0z	16	Circle *[with overlay]*
\1	A1␣	17	
\2	A2␣	18	
\3	A3␣	19	
\4	A4␣	20	
\5	A5␣	21	
\6	A6␣	22	
\7	A7␣	23	
\8	A8␣	24	
\9	A9␣	25	Gas Station (blue pump)
\:	NR␣	26	Hail
\;	NS␣	27	Park/Picnic Area
\<	NT␣	28	NWS Advisory (gale flag)
\=	NU␣	29	
\>	NVz	30	Car *[with overlay]*
\?	NW␣	31	Information Kiosk (blue box with ?)
\@	NX␣	32	Hurricane/Tropical Storm
\A	AAz	33	Box *[with overlay]*
\B	AB␣	34	Blowing Snow
\C	AC␣	35	Coastguard

APRS SYMBOL TABLES (continued)

(Each highlighted character in the Alternate Symbol Table may be replaced with an overlay character).

PRIMARY SYMBOL TABLE			
/$	***GPS xyz***	***GPS Cnn***	***Icon***
/D	PD␣	36	
/E	PE␣	37	Eyeball (eye catcher)
/F	PF␣	38	
/G	PG␣	39	Grid Square (6-character)
/H	PH␣	40	Hotel (blue bed icon)
/I	PI␣	41	TCP/IP
/J	PJ␣	42	
/K	PK␣	43	School
/L	PL␣	44	
/M	PM␣	45	MacAPRS
/N	PN␣	46	NTS Station
/O	PO␣	47	Balloon (SSID –11)
/P	PP␣	48	Police
/Q	PQ␣	49	
/R	PR␣	50	Recreational Vehicle (SSID –13)
/S	PS␣	51	Space Shuttle
/T	PT␣	52	SSTV
/U	PU␣	53	Bus (SSID –2)
/V	PV␣	54	ATV
/W	PW␣	55	National Weather Service Site
/X	PX␣	56	Helicopter (SSID –6)
/Y	PY␣	57	Yacht (sail boat) (SSID –5)
/Z	PZ␣	58	WinAPRS
/[	HS␣	59	Jogger
/\	HT␣	60	Triangle (DF)
/]	HU␣	61	PBBS
/^	HV␣	62	Large Aircraft
/_	HW␣	63	Weather Station (blue)
/`	HX␣	64	Dish Antenna
/a	LA␣	65	Ambulance (SSID –1)
/b	LB␣	66	Bicycle (SSID –4)
/c	LC␣	67	
/d	LD␣	68	Dual Garage (Fire Department)
/e	LE␣	69	Horse (equestrian)
/f	LF␣	70	Fire Truck (SSID –3)

ALTERNATE SYMBOL TABLE			
\$	***GPS xyz***	***GPS Enn***	***Icon***
\D	AD␣	36	Drizzle
\E	AE␣	37	Smoke
\F	AF␣	38	Freezing Rain
\G	AG␣	39	Snow Shower
\H	AH␣	40	Haze
\I	AI␣	41	Rain Shower
\J	AJ␣	42	Lightning
\K	AK␣	43	Kenwood
\L	AL␣	44	Lighthouse
\M	AM␣	45	
\N	AN␣	46	Navigation Buoy
\O	AO␣	47	
\P	AP␣	48	Parking
\Q	AQ␣	49	Earthquake
\R	AR␣	50	Restaurant
\S	AS␣	51	Satellite/PACsat
\T	AT␣	52	Thunderstorm
\U	AU␣	53	Sunny
\V	AV␣	54	VORTAC Nav Aid
\W	AWz	55	NWS Site *[with overlay]*
\X	AX␣	56	Pharmacy Rx
\Y	AY␣	57	
\Z	AZ␣	58	
\[	DS␣	59	Wall Cloud
\\	DT␣	60	
\]	DU␣	61	
\^	DVz	62	Aircraft *[with overlay]*
_	DWz	63	WX Stn with digi (green) *[w/ ov'lay]*
\`	DX␣	64	Rain
\a	SAz	65	(**A**=ARRL, **R**=RACES etc) *[w/ ov'lay*
\b	SB␣	66	Blowing Dust/Sand
\c	SCz	67	Civil Defense (RACES) *[w/ overlay]*
\d	SD␣	68	DX Spot (from callsign prefix)
\e	SE␣	69	Sleet
\f	SF␣	70	Funnel Cloud

APRS SYMBOL TABLES (continued)

(Each highlighted character in the Alternate Symbol Table may be replaced with an overlay character).

PRIMARY SYMBOL TABLE			
/$	GPS xyz	GPS Cnn	Icon
/g	LG␣	71	Glider
/h	LH␣	72	Hospital
/i	LI␣	73	IOTA (Island on the Air)
/j	LJ␣	74	Jeep (SSID –12)
/k	LK␣	75	Truck (SSID –14)
/l	LL␣	76	
/m	LM␣	77	Mic-repeater
/n	LN␣	78	Node
/o	LO␣	79	Emergency Operations Center
/p	LP␣	80	Rover (puppy dog)
/q	LQ␣	81	Grid Square shown above 128m
/r	LR␣	82	Antenna
/s	LS␣	83	Ship (power boat) (SSID –8)
/t	LT␣	84	Truck Stop
/u	LU␣	85	Truck (18-wheeler)
/v	LV␣	86	Van (SSID –15)
/w	LW␣	87	Water Station
/x	LX␣	88	X-APRS (Unix)
/y	LY␣	89	Yagi at QTH
/z	LZ␣	90	
/{	J1␣	91	
/|	J2␣	92	*[Reserved — TNC Stream Switch]*
/}	J3␣	93	
/~	J4␣	94	*[Reserved — TNC Stream Switch]*

ALTERNATE SYMBOL TABLE			
\$	GPS xyz	GPS Enn	Icon
\g	SG␣	71	Gale Flags
\h	SH␣	72	Ham Store
\i	SIz	73	Indoor short range digi *[w/ overlay]*
\j	SJ␣	74	Work Zone (steam shovel)
\k	SK␣	75	
\l	SL␣	76	Area Symbols (box, circle, etc)
\m	SM␣	77	Value Signpost {3-char display}
\n	SNz	78	Triangle *[with overlay]*
\o	SO␣	79	Small Circle
\p	SP␣	80	Partly Cloudy
\q	SQ␣	81	
\r	SR␣	82	Restrooms
\s	SSz	83	Ship/Boat (top view) *[with overlay]*
\t	ST␣	84	Tornado
\u	SUz	85	Truck *[with overlay]*
\v	SVz	86	Van *[with overlay]*
\w	SW␣	87	Flooding
\x	SX␣	88	
\y	SY␣	89	
\z	SZ␣	90	
\{	Q1␣	91	Fog
\\|	Q2␣	92	*[Reserved — TNC Stream Switch]*
\}	Q3␣	93	
\~	Q4␣	94	*[Reserved — TNC Stream Switch]*

APPENDIX 3: 7-BIT ASCII CODE TABLE

In addition to listing the ASCII character codes in their usual form, this table also expresses the hexadecimal codes for the ASCII digits 0–9 and the upper-case letters A–Z in *shifted* form; i.e. shifted one bit left. This is particularly useful for decoding callsigns and Mic-E position information contained in the address fields of AX.25 frames.

Part 1: Codes 0–31 decimal (00–1f hexadecimal)

Dec	Hex	Char		
0	00	NUL	CTRL-@	
1	01	SOH	CTRL-A	Start of Header
2	02	STX	CTRL-B	Start of Text
3	03	ETX	CTRL-C	End of Text
4	04	EOT	CTRL-D	End of Transmission
5	05	ENQ	CTRL-E	Enquiry (Poll)
6	06	ACK	CTRL-F	Acknowledge
7	07	BEL	CTRL-G	Bell
8	08	BS	CTRL-H	Backspace
9	09	HT	CTRL-I	Horizontal Tab
10	0a	LF	CTRL-J	Line Feed
11	0b	VT	CTRL-K	Vertical Tab
12	0c	FF	CTRL-L	Form Feed
13	0d	CR	CTRL-M	Carriage Return
14	0e	SO	CTRL-N	Shift Out
15	0f	SI	CTRL-O	Shift In
16	10	DLE	CRTL-P	Data Link Escape
17	11	DC1/XON	CTRL-Q	Device Control 1
18	12	DC2	CTRL-R	Device Control 2
19	13	DC3/XOFF	CTRL-S	Device Control 3
20	14	DC4	CTRL-T	Device Control 4
21	15	NAK	CTRL-U	Negative Acknowledge
22	16	SYN	CTRL-V	Synchronous Idle
23	17	ETB	CTRL-W	End of Transmission Block
24	18	CAN	CTRL-X	Cancel
25	19	EM	CTRL-Y	End of Medium
26	1a	SUB	CTRL-Z	Substitute
27	1b	ESC	CTRL-[	Escape
28	1c	FS	CTRL-\	File Separator
29	1d	GS	CTRL-]	Group Separator
30	1e	RS	CTRL-^	Record Separator
31	1f	US	CTRL-_	Unit Separator

Part 2: Codes 32–127 decimal (20–7f hexadecimal), including hex codes for shifted 0–9/A–Z

Dec	Hex	Char	Shifted
32	20	␣	40/41 (space)
33	21	!	
34	22	"	(inv commas)
35	23	#	
36	24	$	
37	25	%	
38	26	&	
39	27	'	(apostrophe)
40	28	(	
41	29	)	
42	2a	*	
43	2b	+	
44	2c	,	(comma)
45	2d	-	(minus)
46	2e	.	(dot)
47	2f	/	
48	30	0	60/61
49	31	1	62/63
50	32	2	64/65
51	33	3	66/67
52	34	4	68/69
53	35	5	6a/6b
54	36	6	6c/6d
55	37	7	6e/6f
56	38	8	70/71
57	39	9	72/73
58	3a	:	
59	3b	;	
60	3c	<	
61	3d	=	
62	3e	>	
63	3f	?	
64	40	@	
65	41	A	82/83
66	42	B	84/85
67	43	C	86/87
68	44	D	88/89
69	45	E	8a/8b
70	46	F	8c/8d
71	47	G	8e/8f
72	48	H	90/91
73	49	I	92/93
74	4a	J	94/95
75	4b	K	96/97
76	4c	L	98/99
77	4d	M	9a/9b
78	4e	N	9c/9d
79	4f	O	9e/9f
80	50	P	a0/a1
81	51	Q	a2/a3
82	52	R	a4/a5
83	53	S	a6/a7
84	54	T	a8/a9
85	55	U	aa/ab
86	56	V	ac/ad
87	57	W	ae/af
88	58	X	b0/b1
89	59	Y	b2/b3
90	5a	Z	b4/b5
91	5b	[	
92	5c	\	
93	5d	]	
94	5e	^	
95	5f	_	(underscore)
96	60	`	(grave accent)
97	61	a	
98	62	b	
99	63	c	
100	64	d	
101	65	e	
102	66	f	
103	67	g	
104	68	h	
105	69	i	
106	6a	j	
107	6b	k	
108	6c	l	
109	6d	m	
110	6e	n	
111	6f	o	
112	70	p	
113	71	q	
114	72	r	
115	73	s	
116	74	t	
117	75	u	
118	76	v	
119	77	w	
120	78	x	
121	79	y	
122	7a	z	
123	7b	{	
124	7c	|	
125	7d	}	
126	7e	~	
127	7f	DEL	

APPENDIX 4: DECIMAL-TO-HEX CONVERSION TABLE

Dec	Hex
128	80
129	81
130	82
131	83
132	84
133	85
134	86
135	87
136	88
137	89
138	8a
139	8b
140	8c
141	8d
142	8e
143	8f
144	90
145	91
146	92
147	93
148	94
149	95
150	96
151	97
152	98
153	99
154	9a
155	9b
156	9c
157	9d
158	9e
159	9f

Dec	Hex
160	a0
161	a1
162	a2
163	a3
164	a4
165	a5
166	a6
167	a7
168	a8
169	a9
170	aa
171	ab
172	ac
173	ad
174	ae
175	af
176	b0
177	b1
178	b2
179	b3
180	b4
181	b5
182	b6
183	b7
184	b8
185	b9
186	ba
187	bb
188	bc
189	bd
190	be
191	bf

Dec	Hex
192	c0
193	c1
194	c2
195	c3
196	c4
197	c5
198	c6
199	c7
200	c8
201	c9
202	ca
203	cb
204	cc
205	cd
206	ce
207	cf
208	d0
209	d1
210	d2
211	d3
212	d4
213	d5
214	d6
215	d7
216	d8
217	d9
218	da
219	db
220	dc
221	dd
222	de
223	df

Dec	Hex
224	e0
225	e1
226	e2
227	e3
228	e4
229	e5
230	e6
231	e7
232	e8
233	e9
234	ea
235	eb
236	ec
237	ed
238	ee
239	ef
240	f0
241	f1
242	f2
243	f3
244	f4
245	f5
246	f6
247	f7
248	f8
249	f9
250	fa
251	fb
252	fc
253	fd
254	fe
255	ff

APPENDIX 5: GLOSSARY

Altitude	1. In Mic-E format, the altitude in meters relative to 10km below mean sea level.
	2. In Comment text, the altitude in feet above mean sea level.
Announcement	An APRS message that is repeated a few times an hour, perhaps for several days.
Announcement Identifier	A single letter A-Z that identifies a particular announcement.
Antenna Height	In NMEA sentences, the height of the antenna in meters relative to mean sea level. (The antenna height in GPS NMEA sentences fluctuates wildly because of Selective Availability, and should only be used if DGPS correction is applied).
APRS	Automatic Position Reporting System.
APRS Data	The data that follows the APRS Data Type Identifier in the AX.25 Information field and precedes the APRS Data Extension.
APRS Data Extension	A 7-byte extension to APRS Data. The Data Extension includes one of Course/Speed, Wind Direction/Wind Speed, Station Power/Antenna Effective Height/Gain/Directivity, Pre-Calculated Radio Range, DF Signal Strength/Effective Antenna Height/Gain, Area Object Descriptor.
APRS Digipeater Path	A digipeater path via repeaters with RELAY, WIDE and related aliases. Used in Mic-E compressed location format.
APRS Data Type Identifier	The single-byte identifier that specifies what kind of APRS information is contained in the AX.25 Information field.
Area Object	A user-defined graphic object (circle, ellipse, triangle, box and line).
ASCII	American Standard Code for Information Interchange. A 7-bit character code conforming to ANSI X3.4 (1968) — see Appendix 3 for character definitions.
AX.25	Amateur Packet-Radio Link-Layer Protocol.
Base 91	Number base used to ensure that numeric values are transmitted as printable ASCII characters. To obtain the character string corresponding to a numeric value, divide the value progressively by decreasing powers of 91, and add 33 decimal to the result at each step. Printable characters are in the range **!**..**{**. Used in compressed lat/long and altitude computation.
Bulletin	An APRS message that is repeated several times an hour, for a small number of hours. A General Bulletin is addressed to no-one in particular. A Group Bulletin is addressed to a named group (e.g. WX).
Bulletin Identifier	A single digit 0-9 that identifies a particular bulletin.
Destination Address field	The AX.25 Destination Address field, which can contain an APRS destination callsign or Mic-E encoded data.
DF Report	A report containing DF bearing and range.
DGPS	Differential GPS. Used to overcome the errors arising from Selective Availability.
DHM	7-character timestamp: day-of-the-month, hour, minute, zulu or local time.
DHMz	7-character timestamp: day-of-the-month, hour, minute, zulu only.
Digipeater	A station that relays AX.25 packets. A chain of up to 8 digipeaters may be specified.
Digipeater Addresses field	The AX.25 field containing 0–8 digipeater callsigns (or aliases).
Directivity	The favored direction of an antenna. Used in the PHG Data Extension.
DX Cluster	A network host that collects and disseminates user reports of DX activity.
ECHO	A generic APRS digipeater callsign alias, for an HF digipeater.
Effective Antenna Height	The height of an antenna above the local terrain (not above sea level). A first-order indicator of the antenna's effectiveness in the local area. Used in the PHG Data

	Extension.
ERP	Effective Radiated Power. Used in Status Reports containing Beam Heading and Power data (typically for meteor scatter use).
FCS	Frame Check Sequence. A sequence of 16 bits that follows the AX.25 Information field, used to verify the integrity of the packet.
GATE	A gateway between HF and VHF APRS networks. Used primarily to relay long-distance HF APRS traffic onto local VHF networks.
GGA Sentence	A standard NMEA sentence, containing the receiving station's lat/long position and antenna height relative to mean sea level, and other data.
GLL Sentence	A standard NMEA sentence, containing the receiving station's lat/long position and other data.
GMT	Greenwich Mean Time (=UTC=zulu).
GPS	Global Positioning System. A global network of 24 satellites that provide lat/long and antenna height of a receiving station.
GPSxyz	An APRS destination callsign that specifies a display symbol from either the Primary Symbol Table or the Alternate Symbol Table. Some symbols from the Alternate Symbol Table can be overlaid with a digit or a letter. Used by trackers that cannot specify the symbol in the AX.25 Information field.
GPSCnn	An APRS destination callsign that specifies a display symbol from the Primary Symbol Table. The symbol can not be overlaid. Used by trackers that cannot specify the symbol in the AX.25 Information field.
GPSEnn	An APRS destination callsign that specifies a display symbol from the Alternate Symbol Table. The symbol can not be overlaid. Used by trackers that cannot specify the symbol in the AX.25 Information field.
HMS	1. In NMEA sentences, a 6-character timestamp: hour, minute, second UTC. 2. In APRS Data, a 7-character timestamp: hour, minute, second, zulu or local.
ICQ	International CQ chat.
IGate	A gateway between a VHF and/or HF APRS network and the Internet.
Information field	The AX.25 Information field containing APRS information.
Item	A type of display object.
Item Report	A report containing the location of an APRS Item.
Killed Object	An Object that an APRS user has assumed control of.
knots	International nautical miles per hour.
KPC-3	A Terminal Node Controller from Kantronics Co Inc.
Longitude Offset	An offset of +100 degrees longitude (used in Mic-E longitude computation).
LORAN	Long Range Navigation System (a terrestrial precursor to GPS).
Maidenhead Locator	A 4- or 6-character grid locator specifying a station's position.
MDHM	8-byte timestamp: month, day, hour, minute (used in positionless weather station reports).
Message	A one-line text message addressed to a particular station.
Message Acknowledgement	An optional acknowledgement of receipt of a message.
Message Group	A user-defined group to receive messages.
Message Identifier	A 1–5 character message identifier (typically a line number).
Mic-E	Originally Microphone Encoder, a unit that encodes location, course and speed information into a very short packet, for transmission when releasing the microphone PTT button. The Mic-E encoding algorithm is now used in other devices (e.g. in the

	PIC-E and the Kenwood TH-D7/TM-D700 radios).
Mic-E Message Identifier	A 3-bit identifier (A/B/C) specifying a standard Mic-E message or custom message code.
Mic-E Message Code	A 3-bit code specifying a Standard or Custom Mic-E message.
MIM	Micro Interface Module. A complete telemetry TNC transmitter on a chip.
mph	miles per hour.
Net Cycle Time	The time within which it should be possible to gain the complete picture of APRS activity (typically 10, 20 or 30 minutes, depending on the number of digipeaters traversed and local conditions). Stations should not transmit status or position information more frequently unless mobile, or in response to a Query.
NMEA	National Marine Electronic Association (United States). Producer of the *NMEA 0183 Version 2.0* specification that governs the format of Received Sentences from navigation equipment (such as GPS and LORAN receivers). See Appendix 6 for a reference to NMEA sentence formats.
NMEA (Received) Sentence	The ASCII data stream received from navigation equipment (such as GPS receivers) conforming to the NMEA 0182 Version 2.0 specification. APRS supports five NMEA Sentences: GGA, GLL, RMC, VTG and WPT.
NRQ	Number/Rate/Quality. A measure of confidence in DF Bearing reports.
Null Position	Default position to be reported if the actual position is unknown or indeterminate. The null position is 0° 0' 0" north, 0° 0' 0" west.
NWS	National Weather Service (United States).
Object	A display object that is (usually) not a station. For example, a weather front or a marathon runner.
Object Report	A report containing the position of an object, with optional timestamp and APRS Data Extension.
PHG	APRS Data Extension specifying Power, Effective Antenna Height/Gain/Directivity.
PIC	Programmable Interface Controller.
PIC-E	A PIC implementation of the Mic-E microphone encoder.
Position Ambiguity	A reduction in the accuracy of APRS position information (implemented by replacing low-order lat/long digits with spaces). Used when the exact position is not known.
Position Report	A report containing lat/long position, optionally with timestamp and Data Extension.
Pre-Calculated Radio Range	A station's estimate of omni-directional radio range (in miles). Used in compressed lat/long format.
Query	A request for information. Queries may be addressed to stations in general or to specific stations.
Range Circle	Usable radio range (in miles), computed from PHG data.
RELAY	A generic APRS digipeater callsign alias, for a VHF/UHF digipeater with limited local coverage.
Response	A reply to a query.
RMC Sentence	A standard NMEA sentence, containing the receiving station's lat/long position, course and speed, and other data.
RTCM	Radio Technical Commission for Maritime Services. The RTCM SC104 data format specification describes the requirements for differential GPS data correction.
Selective Availability	Deliberate GPS position dithering, introducing significant received position errors in latitude, longitude *and* antenna height. Errors can be greatly reduced with differential GPS.
Sentence	See NMEA (Received) Sentence.
Signpost	A special signpost icon that displays user-defined variable information (such as a

	speed limit or mileage) as an overlay.
Skywarn	A weather spotter initiative coordinated by the United States National Weather Service.
Source Address Field	The AX.25 Source Address field, containing the callsign of the originating station. A non-zero SSID specifies a display symbol.
Source Path Header	The digipeater path followed prior to a packet entering a Third-Party Network.
SPCL	A generic APRS destination callsign used for special stations.
SSID	Secondary Station Identifier. A number in the range 0-15, as an adjunct to an AX.25 address. If the SSID in a source address is non-zero, it specifies a display symbol. (This is used when the station is unable to specify the symbol in the AX.25 Destination Address field or Information field).
Station Capabilities	A list of station characteristics that is sent in reply to a query.
Status Report	A report containing station status information (and optionally a Maidenhead locator).
Switch Stream Character	A character normally used for switching TNC channels.
Symbol	A display icon. Consists of a Symbol Table Identifier/Symbol Code pair. Generically, **/$** represents a symbol from the Primary Symbol Table, and **\$** represents a symbol from the Alternate Symbol Table.
Symbol Code	A code for a symbol within a Symbol Table.
Symbol Table Identifier	An ASCII code specifying the Primary Symbol Table (**/**) or Alternate Symbol Table (****). The Symbol Table Identifier is also implicit in GPSCnn and GPSEnn destination callsigns.
Target Footprint	A target area for queries. The querying station asks for responses from stations within a specified number of miles of a lat/long position.
TH-D7	A combined VHF/UHF handheld radio and APRS-compatible TNC from Kenwood.
TM-D700	A combined VHF/UHF mobile radio and APRS-compatible TNC from Kenwood.
Third Party Network	A non-APRS network that does not understand AX.25 addresses (e.g. the Internet).
Third-Party Header	A Path Header with the Third-Party Network Identifier and the callsign of the receiving gateway inserted.
TNC	Terminal Node Controller. A combined AX.25 packet assembler/disassembler and modem.
Trace	An APRS query that asks for the route taken by a packet to a specified station.
TRACE	A generic digipeater callsign alias, for digipeaters that performs callsign substitution. These digipeaters self-identify packets they digipeat, by inserting their own callsign in place of RELAY,WIDE or TRACE.
Tracker	A unit comprising a GPS receiver (to obtain the current geographical position) and a radio transmitter (to transmit the position to other stations).
Tunneling	Passing APRS AX.25 traffic through a third-party network that does not understand AX.25 addressing. The AX.25 addresses are carried as data (in the Source Path Header) through the tunneled network.
UI-Frame	AX.25 Unnumbered Information frame. APRS uses only UI-frames — that is, it operates entirely in connectionless (UNPROTO) mode.
UNPROTO Path	The digipeater path to the destination callsign.
UTC	Coordinated Universal Time (=zulu=GMT).
VTG Received Sentence	A standard NMEA sentence, containing the receiving station's course and speed.
WIDE	A generic APRS digipeater callsign alias, for a digipeater with wide area coverage.
WIDEn-N	A generic APRS digipeater callsign alias, for a digipeater with wide area coverage (N=0-7). As a packet passes through a digipeater, the value of N is decremented by 1 until it reaches zero. The digipeater keeps a record of each packet (or its FCS) as it

passes through, and will not digipeat the packet again if it has digipeated it already within the last 28 seconds.

WPT Sentence A standard NMEA sentence, containing waypoints.

WX Weather.

Ziplan A cheap twisted-pair LAN connecting PCs via their serial I/O ports. Designed for use in emergency situations.

Zulu UTC/GMT.

Units Conversion Table

To convert from	*to*	*multiply by*	*divide by*
feet	meters	0.3048	
meters	feet		0.3048
miles	km	1.609344	
km	miles		1.609344
miles	nautical miles	0.8689762	
nautical miles	miles		0.8689762
miles per hour (mph)	knots	0.8689762	
knots	miles per hour (mph)		0.8689762
knots	meters / second	0.51444’	
meters / second	knots		0.51444’
miles per hour (mph)	meters / second	0.44704	
meters / second	miles per hour (mph)		0.44704

Fahrenheit / Celsius Temperature Conversion Equations

$F = (C \times 1.8) + 32$

$C = \frac{(F - 32) \times 5}{9}$

APPENDIX 6: REFERENCES

AX.25 Amateur Packet-Radio Link-Layer Protocol Version 2.0, October 1984, at
http://www.tapr.org/tapr/html/ax25.html

NMEA 0183 ASCII Interface Specification, at http://www.nmea.org/0183.htm

NMEA Sentence Formats, in the *Garmin GPS25 Technical Reference Manual*, at
http://www.garmin.com/manuals/spec25.pdf

Maidenhead Locator, in the *IARU Region 1 VHF Manager's Manual*, at
http://www.scit.wlv.ac.uk/vhfc/iaru.r1.vhfm.4e/index.html

APPENDIX 7: DOCUMENT RELEASE HISTORY

Date	Doc Version	Status / Major Changes
10 Oct 1999	1.0 (Draft)	Protocol Version 1.0. First public draft release.
3 Dec 1999	1.0.1g	Protocol Version 1.0. Second public draft release. Much extended, incorporating packet format layouts, APRS symbol tables, compressed data format, Mic-E format, telemetry format.
30 Apr 2000	1.0.1m	Protocol Version 1.0. Third public draft release. Major additions/changes to the draft 1.0.1g specification: • Added a section on Map Views and Range Scale. • Changed Destination Address SSID description (specifying generic APRS digipeater paths) to apply to *all* packets, not just Mic-E packets. • Changed APRS destination "callsigns" to "destination addresses". • Added `TEL*` to the list of generic destination addresses. • Added brief explanations of how several generic destination addresses are used. • Added "Grid-in-To-Address" (but marked as obsolete). • Extended the description of the Comment field, with pointers to what can appear in the field. • Added explanation of base 91. • Added paragraph on lack of consistency in on-air units, and default GPS datum = WGS84. • APRS Data Type Identifiers Table: marked Shelter Data and Space Weather as reserved DTIs. marked the `-` DTI as unused (previously erroneously allocated to Killed Objects). marked the `'` DTI to mean *Current* Mic-E data in Kenwood TM-D700 radios. marked the `‘` DTI as *not used* in Kenwood TM-D700 radios. • Position Ambiguity: need only be specified in the latitude — the longitude will have the same level of ambiguity. • Added the options of `.../...` and `___/___` to express unknown course/speed. • Added DFS parameter table. • Added Quality table for BRG/NRQ data. • Position, DF and Compressed Report formats: split the format diagrams into two parts (with and without timestamps). • DF Reports: added notes: BRG/NRQ data is only valid when the symbol is `/\`. CSE=000 means the DF station is fixed, CSE non-zero means the station is moving. • Compressed position reports: corrected the multiplication/division constants for encoding/ decoding. • Mic-E chapter rewritten and expanded. Emphasized the need to ensure that non-printing ASCII characters are not dropped. Corrected the Mic-E telemetry data format. • Expanded the introductory description of Objects/Items. All Objects must have a timestamp. • Added Area Object Extended Data field to Object and Item format diagrams. • Added Object/Item format diagrams with compressed location data. • Killed Objects/Items: now indicated by underscore after the name. (continued on the next page)

Date	*Doc Version*	*Status / Major Changes*
	1.0.1m (continued)	• Re-categorized weather reports: Raw, Positionless and Complete. • Added a statement that temperatures below zero are expressed as -01 to -99. • Added the options of `...` and `___` to express unknown weather parameter values. • Corrected the storm data format. Also, central pressure is now `/ppppp` (tenths of millibar). • Corrected the telemetry parameter data (now APRS *messages* instead of AX.25 UI *beacons*). • Added optional comment field to the Telemetry (`T`) format. • Added a section describing the handling of multiple message acknowledgements. • Added a section on NTS radiograms. • Added Bulletin/Announcement implementation recommendations. • Queries and Responses: Query Names (e.g. APRSD): all upper-case. A queried station need not respond if it has no relevant information to send. A queried station should ignore any query type that it does not recognize. APRSH: callsigns must be padded to 9 characters. • Added `PING` as a synonym of `APRST`. • Extended meteor scatter ERP beyond 810 watts, and added a lookup table. • Maidenhead Locator: all letters must be transmitted in upper case, but may be received in either upper or lower case. • Changed the definition of non-APRS packets — these are not APRS Status Messages, but may optionally be treated as such. • APRS Symbols chapter substantially rewritten.. • Added section on Symbol Precedence (where more than one symbol appears in an APRS packet). • Clarified some of the descriptions in the APRS Symbol Tables. • Added overlay capability to the `\a` symbol (ARES/RACES etc). • Separated the 7-bit ASCII table from the Dec/Hex (0x80-0xff) conversion table. • Added several new entries and a units conversion table to the Glossary. • Added new references to NMEA sentence formats and Maidenhead Locator formats.

Date	Doc Version	Status / Major Changes
29 Aug 2000	1.0.1	Protocol Version 1.0. Approved public release. Minor additions/changes to the draft 1.0.1m specification: • Added Foreword. • Replaced section on Map Views and Range Scale. • APRS Software Version No: added **APD**xxx (Linux aprsd server). • APRS Data Type Identifier: Designated **[** as Maidenhead grid locator (but noted as obsolete). • Position Ambiguity: added a bounding box example. • Compressed Position Formats: for course/speed, corrected the range of possible values of the "c" byte to 0–89. • Mic-E: replaced the latitude example table, to show more explicitly how the N/S/E/W/Long offset bits are encoded. • Mic-E: removed the paragraph stating that there must be a space between the altitude and comment text — no space is required. • Mic-E: removed the note on inaccurate altitude data, as GPS Selective Availability has been switched off. • Object Reports: added timestamps to some of the examples (an Object Report must always have a timestamp). • Signposts: can be Objects or Items. • Storm Data: changed central pressure format to **/**pppp (i.e. to the nearest millibar/hPascal). • Storm Data: Hurricane Brenda examples: inserted a leading zero in the central pressure field (central pressure is 4 digits). • Telemetry Data: Added **MIC** as an alternative form of Sequence Number. **MIC** may or may not be followed by a comma. • Messages: added the reject message format. • Appendix 1: Agrelo format: changed the separator between Bearing and Quality to **/**. • Symbol Table: changed **/(** symbol from "Cloudy" to "Mobile Satellite Groundstation". • Reformatted the Units Conversion Table.

END OF DOCUMENT