# Table of Contents

Introduction ......................................................... 3  
  - About NWR .................................................. 3

Project Description ............................................... 5  
  - Product Dimensions and Applications ...................... 5  
  - Functional Hardware Description ......................... 6  
  - Microcontroller Specifics .................................. 7  
  - User Interface .............................................. 8  
  - Power Supply ............................................... 9  
  - Software Description ...................................... 9

Development Plan .................................................. 10  
  - Sequence of Tasks ......................................... 10  
  - Development Hardware/Software ......................... 12  
  - Demonstration ............................................. 12

Electrical Specifications ......................................... 13  
  - Project Specifications .................................... 13  
  - Power Requirements ...................................... 13  
  - Environmental Requirements/PCB size .................. 14

Preliminary Parts List ............................................ 14
Introduction

NOAA (National Oceanic and Atmospheric Administration) Weather Radio (NWR) provides up to date information on local weather forecasts and serves as an emergency broadcasting system that is specific to areas the size of a few counties. It is broadcasted near 162.5 MHz and consists of analog audio messages that are updated approximately every five minutes and a digital Audio Frequency Shift Keyed (AFSK) signal that is transmitted before the analog audio in the event of an emergency. The AFSK message, which is translated to ASCII (American Standard Code for Information Interchange), has its own higher level data format known as Specific Area Message Encoding (SAME). Also, in the event of an emergency, a selective signal is sent before audio transmission to automatically turn on weather radios.

I am proposing to build an AFSK decoder that turns on automatically in the event of an emergency, sounds an alarm, and displays the SAME message on an LCD. The input to the circuit will come from an eighth-inch output audio jack of any weather radio receiver. The Motorola HC9S12 microcontroller will be used to detect the automatic turn-on signal, retrieve the transmitted SAME data, and control the display/alarm functions.

About NWR

NWR stations typically have a transmitting power of about 100 watts so that the broadcast signal doesn’t carry over to another station on the same frequency. If stations were too powerful, Western Washington might get a tornado warning from Nebraska which would be chaos. The frequencies at which NWR is broadcasted in MHz are shown below.

| 162.400 | 162.425 | 162.450 | 162.475 | 162.500 | 162.525 | 162.550 |
For the Whatcom County area, Puget Sound Marine (162.425 MHz) is the station to tune to and is located on the Olympic Peninsula. Keep in mind that this frequency needs to be down-converted to the audio frequency range until it can actually be heard. This work is done by the receiver and not this decoder. Because of the asynchronous and somewhat random nature of emergencies, I am designing this circuit around a pre-recorded emergency broadcasted SAME message that I found on the web.

SAME messages are broadcasted at 530.83 bits/sec which corresponds to a period of 1.92 ms. A logic zero is 1562.5 Hz and a logic one is 2083.3 Hz. The Warning Alarm Tone (WAT) is the tone that is broadcasted to automatically turn on radios and its frequency is 1050 Hz. A typical SAME message includes:

1. Preamble
2. Header Code
3. Warning Alarm Tone
4. Voice Message
5. Preamble
6. End of Message

The Preamble is 16 bytes of hexadecimal AB and is mainly for MCU clock synchronization and setting AGC. The header code is in ASCII format and is in a block form shown below (in symbolic letters of course).

**ZCZC-WXR-EEE-PSSCCC-PSSCCC+TTTT-JJJHHMM-LLLLLLLL-**

Each block of ASCII characters between the dashes corresponds to locations, directions, warning types, etc. The warning alarm tone and voice message may or may not be transmitted and then the preamble is sent again along with the End Of Message (EOM) bytes NNNN.
Project Description

Project Dimensions and Applications

The NOAA SAME decoder will look something like the sketch below.

The product will be hand held so that it can be carried outside and the maximum dimensions will be 4 x 2 x 9 inches. The case will be made of metal or hard plastic to protect the innards. Two switches will be used; one to turn the device on/off and one to select battery/wall supply (N.C. type). There will be a power ‘on’ LED mounted on the front to indicate power and a 1/8 inch input jack. To use the decoder the user simply needs to turn on the receiver, connect the receiver to the decoder, and turn on the power. It is important to note that the decoder will only display messages on the LCD screen. If the user wants to listen to the voice audio, he or she can only
listen to it through the receiver speaker by disconnecting it from the decoder. The decoder’s primary use is for receivers that don’t have SAME technology.

Functional Hardware Description

There are many different techniques for demodulating FSK signals. I have chosen a tone decoder (PLL type) method that will lock on to the logic 1 frequency. The WAT signal is detected in a similar way. The functional block diagram for the detecting circuitry along with the HC9S12 peripherals are shown below.

Functional Block Diagram

The Band Pass Filters (BPF) will detect the Warning Alarm Tone (1025 Hz) and the logic one (2083.5 Hz) signals and amplify them. Both signals will pass through tone decoders tuned to the appropriate tones. Both outputs of the tone decoders will be connected to ports of the HC9S12.
along with the Alarm and LCD module. The alarm will be of piezo-electric material and connect to the microcontroller through an FET.

Microcontroller Specifics

I will be using the Motorola MC9S12DP256B, although I envision using a much cheaper and more simplistic microcontroller for the final product. A small eight pin HC08 might do just fine so long as it has a sufficient amount of Flash ROM. A functional diagram of the HC9S12 resources being used is shown below along with the power supply block diagram.

Microcontroller Resource Diagram
The -PSSCCC- section of the SAME message contains Federal Information Processing System (FIPS) codes that correspond to specific states and counties. This means I need a string of characters that correspond to each state/county stored in Flash ROM. If I don’t have enough memory to do this, I will only program in northwestern states. In addition to state and county string arrays, I will need space in the Flash ROM for all types of warning messages and what branch of the government sent them. The RAM will be used for global variables and to store the burst of SAME data (268 bytes max). The WAT tone, Logic 1 signal, and Alarm will be assigned to T0, T1 and T2 respectively. The LCD will use A0-A7 of Port A and K0-K2 of Port K. Note that a serial fed LCD could be used instead of a parallel if one wanted to use a more limited microcontroller. A BDM connector will be available for on-chip debugging and memory programming. There will also be a low voltage reset IC (MAXIM) that will automatically reset the microcontroller when supply voltage drops below 4.5 V, and, the batteries will be connected to a voltage regulator because the tone decoder center frequency varies with supply voltage. A 16 MHz crystal will be connected for generation of the e-clock.

User Interface

The display shall be in scrolling message format on a 2x16 backlit LCD screen (turned on only when a signal is detected), with the station that transmitted it stationary above. Below is an example of a typical message.

<table>
<thead>
<tr>
<th>K</th>
<th>B</th>
<th>H</th>
<th>M</th>
<th>/</th>
<th>N</th>
<th>W</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>H</td>
<td>E</td>
<td>R</td>
<td>E</td>
<td>I</td>
<td>S</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>M</td>
</tr>
</tbody>
</table>
The microcontroller will update the LCD so that the bottom line is scrolling and the top fixed. The scrolling message will tell the reader what kind of emergency is present, what areas are affected and how long the message is valid for. The fixed part of the message will display the station that sent the message and who the primary source was. In the example above, KBHM is the station and National Weather Service is the source. This display will turn on only when a valid message has been received in order to save power.

Power Supply

After a few initial estimates, I calculated that my entire circuit would draw no more than 300 mA. I chose a 5 V wall transformer supply with a max current rating of 500 mA just to be safe. For battery powered operation I chose 4 AA Energizer E2 batteries because of their long life and modest cost. The batteries will be regulated with a 5 V, 400 mA regulator (LM2937). The wall transformer power will automatically be used when the 2.5mm plug is connected into the 2.5mm coax jack protruding from the case (N.C switch). When the wall transformer is not connected, battery power will be used. A sliding single-pole single-throw switch will be used to turn all power on or off.

Software Description

The software will be written in C and assembly languages. I will be implementing the MicroC/OS kernel because I will be very familiar with it by the time I start writing software. In addition, I will need to poll for incoming data at least every 0.96 ms, which might cause problems if the decoding or alarm subroutines turn out to be sufficiently large. Because of battery operation I will be setting the microcontroller in wait mode until a valid message has been detected. The major modules for my software are shown on the next page.
Software Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>Polls for SAME data and WAT tone</td>
</tr>
<tr>
<td>LCD</td>
<td>In charge of updating LCD screen with scrolling and stationary text</td>
</tr>
<tr>
<td>DecodeSame</td>
<td>Clocks in the data and creates ASCII message from SAME data</td>
</tr>
</tbody>
</table>

Development Plan

Sequence of Tasks

None of the parts required for this project are hard to find. I should have all of them within a couple of weeks. The main tasks for development of my project are: functional hardware design, PCB board design, software development, and final project construction. The functional hardware design is almost done. This task consisted primarily of bread boarding circuit designs and testing them with a sample signal to check if they work properly. The amplifiers and bandpass filters have already been built. I still need to design the AGC, Alarm, Tone Decoder and change the circuit from dual supply to single supply. The PCB board design will be done on Eagle Layout Editor when I verify that my breadboard design works. Software development will start during winter quarter. The final project construction task includes building a case, securing internal PCB boards, installing the LCD, etc. A schedule showing when these main tasks will be undertaken is shown on the next page.
Winter Quarter

--Week 1-- Finish up Functional Hardware design.

--Week 2-- Finish up Functional Hardware design.

--Week 3-- Start PCB board design. Start learning Eagle Design Software.

--Week 4-- Learn Eagle Design Software.

--Week 5-- Design PCB board layout with Eagle Design Software.

--Week 6-- Continue designing PCB board layout with Eagle Design Software.

--Week 7-- Finish up PCB board layout, start thinking about software.

--Week 8-- Start writing module Main.

--Week 9-- Continue working on Main.

--Week 10-- Continue working on Main.

--Week 11-- Pass finals! Keep working on Main.

--Week 12-- Finish Main. Start LCD module.

Spring Quarter

--Week 1-- Continue working on LCD module.

--Week 2-- Continue working on LCD module, start DecodeSame.

--Week 3-- Work on LCD module, keep working on DecodeSame. Hardware Presentations.

--Week 4-- Finish LCD module, work on DecodeSame.

--Week 5-- Continue working on DecodeSame. Hardware Documents Due.

--Week 6-- Finish DecodeSame. Software System Presentations.

--Week 7-- Start Final Project Construction. Test software and debug.

--Week 8-- Continue Final Project Construction. Test and debug.

Development Hardware/Software

There is quite a lot of equipment that I need to develop the project. I will use most of the bench equipment here in the ET340 and ET333 labs including power supplies, analog/digital oscilloscopes and signal generators to test my hardware. During the Final Product Construction phase I will probably use the metal bending equipment and various tools stored in ET328, and the PCB board routing machine stored in ET342.

For the software phase of development, I plan to program mostly in C and assembly, using the Noral Flex debugger and CodeWright 6.0. I will use Eagle Layout Editor 4.09 for my PCB board and PSPICE for my circuit design. The MicroC/OS kernel will be the backbone of my project’s software.

Demonstration

Hopefully I will have a fully functional FSK modulator/transmitter working within the lab by the time demonstration comes along. That way I can be sure that my circuit is fully functional. I will simulate an actual NWS emergency by transmitting the SAME message three times and then the WAT. To demonstrate that my circuit is insensitive to normal voice audio, I will tune in to the Puget Sound Marine station and let the signal pass through my circuit. If I do not get a FSK modulator/transmitter working, I will test the circuit by connecting it to the ‘speaker out’ of the lab computer’s sound card and play a SAME and WAT sample.

The final prototype will consist of the HC9S12 Eval board connected to a PCB board designed by me, both of which will be connected to the alarm, LCD, switches, and battery pack.
Everything will fit inside of the project case shown in the project sketch. All components on the PCB board will be soldered to the board.

Electrical Specifications

Project Specifications

FCC Regulations (from sec 11.33 & 11.34 EAS of document FCC-02-64A1)
- Stores at least ten pre-selected event and originator header codes
- The display message shall include Originator, Event, Location, and Time
- Shall be readable in normal light and darkness

Input Signal
- Minimum input signal of 75 mV peak
- Maximum input signal of 1 V peak
- Header Codes may not exceed 268 bytes

Power Requirements

Power Dissipation
- Worst-Case Power Emergency Mode, 1.5 watts (300mA)
- Power w/out LCD on and microcontroller in wait mode, 135 mW (30.7 mA)

Source Specifications
- 4 AA Energizer e^2 batteries (3,135 mAh)
- or 5V, 500 mA wall transformer w/2.5mm plug
- life expectancy 15 hours (300 mA), 8 days (30.7 mA)
Environmental Requirements/PCB size

*Operating Temperature Range*

- 32°F --- 120°F

*PCB Maximum Dimensions*

- 2”x3”x1.5” (LxWxH)

### Preliminary Parts List

<table>
<thead>
<tr>
<th>Part</th>
<th>Quantity</th>
<th>Price</th>
<th>Manufacturer</th>
<th>Distributor</th>
<th>Power Dissipation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5V, 500mA Trans</td>
<td>1</td>
<td>$3.50</td>
<td>IBM</td>
<td>All Electronics</td>
<td>2.5 W</td>
</tr>
<tr>
<td>MC9S12DP256B</td>
<td>1</td>
<td>$19.13</td>
<td>Motorola</td>
<td>Motorola</td>
<td>65mA</td>
</tr>
<tr>
<td>LM324 Quad op</td>
<td>1</td>
<td>$0.09</td>
<td>National</td>
<td>National</td>
<td>0.7mA</td>
</tr>
<tr>
<td>Resistors 1%</td>
<td>20</td>
<td>$0.35</td>
<td>Yageo</td>
<td>Digi-Key</td>
<td>5mA</td>
</tr>
<tr>
<td>Capacitors 5%</td>
<td>10</td>
<td>$1.35</td>
<td>Panasonic</td>
<td>Digi-Key</td>
<td>0 W</td>
</tr>
<tr>
<td>LCD 16x2 (backlit)</td>
<td>1</td>
<td>$10.50</td>
<td>Sharp</td>
<td>All Electronics</td>
<td>180mA</td>
</tr>
<tr>
<td>Piezo PKM25-6A0</td>
<td>1</td>
<td>$0.50</td>
<td>Murata</td>
<td>All Electronics</td>
<td>15mA</td>
</tr>
<tr>
<td>Red LED</td>
<td>1</td>
<td>$0.10</td>
<td>Lumex</td>
<td>Digi-Key</td>
<td>10mA</td>
</tr>
<tr>
<td>2.5mm jack/switch</td>
<td>1</td>
<td>$0.25</td>
<td>Sony</td>
<td>All Electronics</td>
<td>0 W</td>
</tr>
<tr>
<td>1/8 inch jack</td>
<td>1</td>
<td>$0.25</td>
<td>Sony</td>
<td>All Electronics</td>
<td>0 W</td>
</tr>
<tr>
<td>Battery Holder AA</td>
<td>1</td>
<td>$0.80</td>
<td>Sony</td>
<td>All Electronics</td>
<td>0 W</td>
</tr>
<tr>
<td>Slide Switch SPST</td>
<td>1</td>
<td>$0.33</td>
<td>Stackpole</td>
<td>All Electronics</td>
<td>0 W</td>
</tr>
<tr>
<td>Transistor</td>
<td>1</td>
<td>$0.27</td>
<td>Fairchild</td>
<td>All Electronics</td>
<td>15mA</td>
</tr>
<tr>
<td>MAX6412</td>
<td>1</td>
<td>$0.50</td>
<td>Maxim</td>
<td>Maxim</td>
<td>4.5uA</td>
</tr>
<tr>
<td>LM567C</td>
<td>2</td>
<td>$0.42</td>
<td>National</td>
<td>Digi-Key</td>
<td>10mA</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>Total Power</strong></td>
<td></td>
<td></td>
<td><strong>1.5 W</strong></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>$38.27</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>