Location Enablement for Advanced Weapons Safety Systems

May 10, 2004

Sponsor: Sandia National Laboratories
Advisors: Dr. Lisa Brown, Dr. Jim Frenzel

Toney Jacobson
jaco1888@uidaho.edu

Devan Williams
will7639@uidaho.edu
Table of Contents

III. Abstract 3
IV. Main Section 4
  1. Project Description 4
    1.1 Background Information 4
    1.2 Problem Statement 5
      1.2.1 Objectives 5
      1.2.2 Requirements 5
      1.2.3 Constraints 5
    1.3 Solution Method 6
      1.3.1 Functions and Means 6
  2. Status 8
    2.1 What is Designed and Working 8
    2.2 What is Designed but Not Working 10
    2.3 What is Designed but Not Tested 11
  3. Method of Solution 12
    3.1 Technical Description 12
      3.1.1 PDA Software - Basic Structure 13
      3.1.2 PDA Software - Forms and Dialog Boxes 14
      3.1.3 PDA Software - Functions 15
      3.1.4 PDA Software - Included Files 15
  4. Validation Procedure 15
    4.1 Test Plan 15
  5. Results 16
    5.1 Operating Procedures 16
    5.2 Validation Results 16
    5.3 Cost Analysis 16
    5.4 Additional Considerations 16
V. Appendix 17
  1. Specifications 17
    1.2 Specifications for each Program Project 17
  2. Bill of Materials 18
  3. PEN-X Controller 18
  4. GPS Module 19
  5. Project Files List 19
  6. Selected Project File Code 19
    6.1 LocationForm.c 29
    6.2 Resources.rcp 34
  7. Palm Tungsten C Data Sheet 2
III. Abstract

Sandia National Laboratories wishes to enhance weapons safety systems by incorporating GPS-monitored location enablement into current systems. The inclusion of a GPS system will allow for a much greater degree of accuracy in the weapons system, which in turn will allow for a much greater degree of safety in the event of an accident involving a hazardous environment to the weapon. In the summer of 2003, we developed a preliminary application of this GPS system, which stored entered target coordinates and characteristics into the software on our system. Government standards require that all target information must be stored in hardware, or entered manually via a user interface with the weapon.

The approach for this project was directly influenced by similar systems within the military which require similar user interfaces. The user interface will be comprised solely of a hand-held Palm-OS based PDA and an infrared receiver. The PDA selected was the Tungsten C model, which comes equipped with both a full keyboard for text entry, and an infrared transmitter. The infrared receiver will be connected serially to the controller board used in the system designed in 2003. The PDA must act as a median between the user and the weapon system it’s associated with. This requires the PDA to accept the user’s target location input, generate parameters based on this input, and then pass them into the previously mentioned controller. The controller will compare the text string output of the GPS system to the received parameters, and create a unique detonation signal based on this comparison. This will allow for either the engagement or the locking of the weapon.

The first half of this project focused on the creation of a program which will allow a user to input target location information. This program was written with Metrowerk CodeWarrior for Palm OS V8.0, and is composed of two main forms, with dialog boxes, buttons, and fields associated with these forms. The buttons on the two main forms bring up pop-up dialog boxes, which contain buttons, checkboxes, and text-entry fields. The user can use the program to select buttons based values associated with the target location (N/S hemisphere, E/W velocity, etc.), and input text-based magnitudes and coordinates (35 k/s S, 125° latitude, etc.) into fields.

Once all the target location information has been input, the PDA will generate parameters based on the inputs, and transmit them to the controller via the infrared transmitter. The design of this infrared transmission and the PDA application will continue through the summer and fall of 2004.
IV. Main Report

1. Project Description

1.1 Background Information

As new technology has become available, Sandia National Laboratories has the opportunity to enhance the safety of weapons safety systems. Current weapons safety systems use trajectory information for enablement of the weapon, but do not take into account location information to enable the weapon. Our project during the summer of 2003 was an initial study to incorporate GPS information into one of the safety subsystems (see Figure 1). This approach is called Location Enablement (LE). Using LE along with intent enablement provides additional benefit to the safety subsystem.

Currently, there are two devices within a weapons safety system that isolate the explosive package of the weapon from the device that produces the energy required to detonate the explosive. These devices, which act as physical barriers to the explosive package, are known as stronglinks. The first stronglink is enabled by password; when the detonation is required, someone must enter the password for detonation in order for the stronglink to open, allowing the energy blast to reach the explosive package of the weapon. This is referred to as intent enablement. The second stronglink is currently enabled by trajectory; the stronglink will open when the weapon experiences an environment similar to the weapon being dropped. Unfortunately, this environment is easily simulated by undesirable circumstances, such as during a crash by the airplane carrying the weapon. The weapon will experience the same acceleration and trajectory as it would if it were dropped. It is because of this that the trajectory process must be improved upon.

The incorporation of location enablement into the second stronglink will increase the safety of the subsystem. Location enablement is a system that determines where the weapon is currently located, and prevents it from detonating anywhere other than the target location. Currently, the target location is stored within the software located on the PEN-X controller (see Figure 2 and the description of the PEN-X controller in the Appendix).
1.2 Problem Statement

Our original system used information provided by a handheld GPS (see Figure 3) to enable the system when the weapon was within the specified range of the target location. This system was a good first step; however, we need to improve the current system for a more realistic and usable system.

The initial system had target location stored in the controller software. For safety concerns, the target location should not be stored on the controller. Sandia National Laboratories requested that the target location be input and reset from a user interface. Our project is the designing and building of an interface box for adding this functionality to the system. The interface box will accept the target location as the inputs and will produce the thresholds of acceptable range as outputs.

1.2.1 Objectives

The primary objective of our project is to design the user interface system that will be incorporated into the existing location enablement system created during the summer of 2003.

1.2.2 Requirements

The requirements for the user interface system are that it must (1) accept target location coordinates as user input, (2) produce the thresholds of acceptable range as output, and (3) store the thresholds of acceptable range in the interface software rather than the controller software.

1.2.3 Constraints

The user interface system must allow a user to enter target location coordinates via a Personal Digital Assistant (PDA) using the Palm Operating System (Palm OS). The PDA software must generate thresholds of acceptable range for the given target location and send the information to a receiver connected to the PEN-X controller (see Figure 2) via an Infrared (IR)
signal. The Infrared receiver must connect to the PEN-X controller via a Serial Port (DB9 RS-232 connection).

1.3 Solution Method

1.3.1 Functions and Means

For this project we must design a user interface system to perform several different functions. The bulk of these functions will take place in software on the Palm PDA. We used a Palm Tungsten C PDA (see Figure 4, and Appendix for more details) for our design. The main functions (described below) will consist of: entering the desired target location, displaying the desired target location to the user for verification, producing the thresholds of acceptable range (from the target location), converting the thresholds into a data string, and sending the thresholds to the PEN-X controller via an Infrared signal.

The target location coordinates must be selected or typed in by the user. Therefore, our system must be able to collect data via a user interface. In order to accomplish this, our PDA software will contain checkboxes for target selection (for hemispheres, velocity directions, and acceleration directions) and text fields for specific location values (velocity magnitudes, acceleration magnitudes, and location coordinates). The details of this are discussed further in the “Status” section.
Our system must then be able to display the data to the user for verification. Therefore, we will have a function that displays all user-entered location values. In case a value was not selected or entered, or an invalid target location was entered, the system must display an error. Therefore, we will have an error-checking function that determines when errors occur. Once the target location data is verified by the user, the thresholds of acceptable range can be produced. These thresholds will be produced from the magnitudes and coordinates typed in by the user, according to predetermined range values.

Our system must also be able to convert the thresholds into data that the PEN-X controller can use. Therefore, we will have a function that converts the target location data into a text string. This string will then be passed on the PEN-X controller for comparison. In order to pass the data on to the PEN-X controller, our software will use the built-in infrared transmitter on the Palm Tungsten C PDA (see Figure 3) to send the data to an Infrared receiver. The Infrared receiver will be connected to the PEN-X controller via a DB9 RS-232 serial port connection. The text string will simply be passed into the PEN-X controller’s built-in RAM buffer for later reference by the PEN-X controller software.
2. Status

2.1 What is Designed and Working

Our design consists of a program written in Palm OS modified C, which, as mentioned before, will accept user inputs for a target location, and generate parameters based on these inputs. Palm OS C coding differs from standard PC C coding in several ways. Our design reflected these differences, and compensated for them in several ways. Palm OS is a visual based operating system, and requires the program to be run in varying “forms”. These forms are basic GUI’s that can be manipulated in several ways. Our design currently consists of two primary forms: the main form, and the location form. The main form is the screen that appears when the program is launched. It consists of two buttons: an “application information” button, and an “enter location information” button. When the application information button is pressed, a pop-up dialog box appears, displaying the program name, the authors, the version, and the date the program was last modified. When the enter location information button is pressed, the program brings up the location form (see Figure 6).
As is seen in Figure 6, the location form consists of nine buttons. The first button is the enter hemispheres/altitude button. When pushed, a pop-up dialog box appears, consisting of three sets of checkboxes, allowing a user to select which hemisphere (N/S, E/W) and what altitude sign (+/-) he wants, and two buttons: a Save button and a Back button (see Figure 7).

Once the user selects the appropriate values (in this case, the Northern hemisphere, the Western hemisphere, and a positive altitude sign), the user has two options. He can either hit the Save button, which stores the values selected as global variables, or he can hit the Back button, which will reset the values to their previously saved values. If the user were to select the Southern hemisphere, and hit the Save button, then whenever he returned to this table, the
Southern hemisphere would be selected. If he selected the Southern hemisphere and hit the *Back* button, then when he returned to this table, the Northern hemisphere would still be selected. A similar dialog box exists for the *Velocity Direction* Button on the *Location Form*. When this button is pushed, a dialog box is displayed that allows the user to input the direction of velocity (using checkboxes similar to Figure 7) along the x-axis (E/W), along the y-axis (N/S) and along the z-axis (U/D).

### 2.2 What is Designed but not Working

Five of the buttons on the Location Form (the *velocity magnitude*, *acceleration magnitude*, and *x-, y-, and z-coordinate* buttons) bring up dialog boxes similar to those discussed previously. One primary difference, however, is that these location entries require more than a binary input. To enter a viable velocity magnitude, the user must have greater freedom when entering inputs. To compensate for this, text-entry fields have been included to replace the checkboxes on the pop-up dialog boxes that appear when a button is selected (see Figure 8).

![Figure 8. The Velocity Magnitude Table.](image)

The gray boxes represent the text entry fields, which will allow a user to input the exact magnitude of a velocity desired. When the *Save* button is selected, these values will be stored, and when the *Back* button is selected, the values will be reset to the original values. The program will store these entered values in global variables, and create parameters of acceptable range based on them. The range will be pre-set, and will be distributed equally over both sides of the entered velocity. This will allow the desired velocity to be in the exact center of the generated parameters, giving it an equal range for error on both sides of the desired velocity.
Similar dialog boxes will be used for the acceleration magnitude, x-coordinate, y-coordinate, and z-coordinate buttons. These dialog boxes will also contain text-entry fields to allow the user to input the exact values he wishes. Parameters will be generated a similar way for the acceleration magnitude inputs. For the x-, y-, and z-coordinate entry, the user will have to enter latitude, longitude, and altitude coordinates in a specific syntax, so the program recognizes them. Parameters will then be generated based on these inputs. All values for field text-entry inputs will be stored as strings, which will be converted into a single string for interpretation by the PEN-X controller when received.

There is one aspect of the text-entry fields that is not working in our design. The program will not allow the user to use the keyboard included on the PDA to enter desired location values in the fields. The program we are using to write the code (Metrowerks Codewarrior for Palm OS V8.0) is responsible for this. Since we had to use a demo version of the software for our project, the software did not allow us to write programs larger than 5 kB in size. As a result, sufficient code could not be written to allow the text-entry fields to function properly. This problem will be addressed directly as our project develops further. To circumvent the problem for the demonstration, text-entry fields were removed, and checkboxes were used which allows the user to select the parameters which normally would have been generated within the software. We have requested from our project sponsor (Lisa Brown, of Sandia National Labs) that the full version of CodeWarrior for Palm OS V9.0 be purchased for us to use when developing our project over the summer. Ms. Brown has agreed to purchase this software, so the problem of not allowing text-entry fields to use the onboard keyboard will be fixed shortly.

2.3 What is Designed but not Tested

Once the PDA receives the user inputs and generates the parameters based on these inputs, it must pass the parameters to the PEN-X controller. In order to do so, the parameters must be output from the PDA via the built-in infrared transmitter. This will require an infrared receiver to be connected to the PEN-X controller. This receiver will connect to one of the many DB9 RS-232 Serial Ports on the PEN-X controller, which has a RAM buffer built-in. Once the Submit button on the main form is pressed, the values entered by the user will be connected together into a single string of values, and the string will be output to the infrared receiver on the
PEN-X controller. The string will be received and immediately stored in the RAM buffer, until the PEN-X controller executes.

When this occurs, the PEN-X controller software will compare the actual location (from the GPS output) to the parameters stored in the buffer. Depending on whether or not the actual location is in a parameter, the controller will output a 1 or a 0 to the stronglink. Once all 24 events are executed, the stronglink will either enable or it will have been locked during the execution, depending on the sequence of 1’s and 0’s received from the PEN-X controller.

This design has yet to be tested, as the main focus of the first semester of design was the compilation of the code necessary to receive user target inputs. As soon as construction of the device which is to receive the infrared signal from the PDA begins, testing will ensue. The PEN-X controller has been tested in previous work to be able to correctly store parameters in the RAM buffer, so the most difficult aspect of designing the infrared system will be to get the infrared devices properly configured to communicate with each other.

3. Method of Solution
3.1 Technical Description

For the development of the Palm OS application, we have been using the development environment “Metrowerks CodeWarrior for Palm OS V8.0” software. This builder, linker, and compiler program takes C programming language code and generates a Palm OS application file. The file generated contains all the necessary application details in a “.prc” file that is ready to be loaded on to the Palm PDA or onto Palm OS Emulator software. We have been using the Palm OS Emulator V3.5 to test our application, because it is more convenient to load the application into the emulator software rather than the PDA itself. The emulator software is simply a program on the computer that acts as a Palm PDA device running under Palm OS. The emulator comes with the CodeWarrior for Palm OS V8.0, or it can be downloaded for free from the Palm website.

We have also been using the book “Palm OS Programming: The Developer’s Guide (2nd Edition),” by Neil Rhodes, for help with Palm OS programming. Coding for the Palm OS environment is quite different from coding for a typical personal computer, as previously mentioned. Most coding on the Palm OS is graphically and spatially oriented, as the above book illustrates, requiring much more attention to detail than typical C coding.
Without going into too much detail about the Palm OS, there are several important details that should be mentioned. The Palm OS is an “event-driven” system. Every screen tap, button press, and keypad press produce an event that must be handled by the system, the menu, or the application itself. Each screen (called a “form” on the Palm OS) requires its own function for event handling, and is typically contained in a separate “<form name> .c” file in the application design project. Each pop-up dialog box requires its own function as well, typically contained in the file of the form it “pops-up” from. The physical layouts of each form and dialog box are described in a “Resources.rcp” file. An example of this type of file is shown in Figure 9. This file will be described later in the “Included Files” section.

**Figure 9. Location Form Layout (left) and its Resource file (right).**

### 3.1.1 PDA Software - Basic Structure

The basic structure of the code for the location application is as follows. Each “screen” in the application is referred to as a “form” and contains buttons and menus. There are two forms within the program: the **Main** form and the **Location** form. The **Main** form consists of three “buttons,” which, when selected, either brings up a pop-up dialog box or transfers the user to another form. These buttons are: “Application Information,” “Enter Location Information,”
and “Results.” Some of the more important files from the PDA software project are located in the Appendix.

3.1.2 PDA Software - Forms and Dialog Boxes

The “Application Information” button displays the authors of the code, the version of the code, and the date the code was last altered on an information table. The “Enter Location Information” button brings up the Location form. The “Results” button shows the location information entered by the user, if any. The flow chart is shown in Figure 10 below.

The Location form consists of nine different buttons (shown in Figure 9): Hemispheres/Altitude, Velocity Direction, Velocity Magnitude, Acceleration Magnitude, X-Coordinate Entry, Y-Coordinate Entry, Z-Coordinate Entry, Submit, and Return.

The Hemispheres/Altitude button brings up a dialog box with the possible hemispheres and altitude of the target (N/S, E/W, +/-). The Velocity Direction button brings up a similar dialog box, with the possible velocity directions (N/S, E/W, U/D).

The first version of our code brought up similar dialog boxes for the other buttons. This allowed the user to select a range for the target magnitudes and coordinates. The second version of our code will bring up dialog boxes with fields for exact text entry of desired magnitudes or coordinates.

![Figure 10. PDA Software Flow Chart](image-url)
3.1.3 PDA Software - Functions

Several Functions are used in the PDA C Code. The function “DisplayHemForm” displays the Hemispheres/Altitude dialog box from the Location Form. Other functions that do similar tasks include: DisplayVelDirForm, DisplayVelMagForm, etc. The function “SetFieldTextFromHandle” allows the user to select which field in a particular dialog box he/she would like to type in. The “SetFieldTextFromString” function allows the user to enter text into a field via the on-PDA keyboard. The “LocationFormHandleEvent” function determines what happens on each button press while in the location form.

3.1.4 PDA Software - Included Files

The header files perform the same function as header files in C and C++. The file “ResourceDefines.h” contains the ID’s and addresses for all forms, buttons and checkboxes within the program. Addresses assigned are four digit numbers. The file “Resources.rcp” contains the definitions for all the forms within the program, specifying size, location, location of objects within form, and ID’s for forms (as shown in Figure 9). The full list of included files is located in the Appendix. Some of the code is also provided in the Appendix.

4. Validation Procedure

4.1 Test Plan

The system of metrics used to test our software is simple. Initially, we assumed the “if it works, then it works” philosophy would be sufficient enough to test the software we developed. It was pointed out, however, that there was no immediate method to make sure the software rejected errors. We then included a test procedure known as “error seeding”, or intentionally entering nonsense values into the software. For instance, for the entry of the x-coordinate, we would error seed the input value “FJOWE”, which has no relevant meaning in terms of latitude. The program will need to recognize this, and output an error on the screen prior to these values being passed into the PEN-X controller. Other methods of error seeding include not selecting a value on the hemisphere (N/S) entry table. The program will need to check to make sure all input values have been entered before proceeding to submit the parameters based on these values. Other metrics will need to be developed to incorporate the testing of the infrared devices, once they are included in the system. Since propagation delay between events is not an issue with this system (all values get passed in at once), no metrics were designed to test the speed of the PDA.
5. Results

5.1 Operating Procedures

Operating the user input application system is pretty self-explanatory. To install the application on the PDA, simply HotSync the application to the PDA using the Palm desktop software provided with the Palm PDA. To run the application on the PDA, just select the “Location” icon from the main application screen on the PDA. Then, to enter location information, tap the “Enter Location Information” button. Tapping this button will open the “Location Information” form. From this form, select the desired button for location information. A detailed explanation of the layout can be found in the “Status” section of this report.

When the user has finished entering the target location values, he can select the Submit button to view the entered locations (not yet completed).

5.2 Validation Results

At this point the application runs as intended. The simplicity of the application keeps it error free, but it also keeps it from being fully functional. At this point there is no way to enter exact velocity magnitudes or location coordinates, due to the limited file size problem discussed earlier.

5.3 Cost Analysis

The sponsor of the project, Sandia National Labs, has provided all the necessary parts, programs, and resources for the project construction. The cost for each component is outlined in section 3 of the Appendix. Sandia has committed to continue to sponsor this project throughout the project lifecycle. The cost for the programs and parts is $1,390.00 which includes the Palm Tungsten C PDA’s, the CodeWarrior software, an Infrared receiver, a printed circuit board, and a Palm OS programming book.

5.4 Additional Considerations

The sole intent for this project is to increase the safety of our nuclear arsenal. There will be no additional safety constraints present, and there will be no safety concerns created by the development of this project. Unfortunately, there are political issues associated with any nuclear-related material. Some people might see this project as a means of increasing the power and potency of nuclear weapons. It is because of this that we must be cognizant in how we refer to our project, and must continually stress the importance of the safety aspect and reasoning behind it.
V. Appendix

1. Specifications

As mentioned in our design review, a Palm OS based PDA must be used to accomplish our objectives. This PDA must have sufficient enough memory space to contain the application we write. For security reasons, the PDA itself must have any means of recording audio, video, or visual captures. The application to be stored on it must accept user location information inputs, and generate acceptable parameters based on these inputs. It must output these parameters via an infrared transmitter to an infrared receiver, which will be connected to the PEN-X controller. The software must allow the user to input the exact velocity and acceleration magnitudes and latitude/longitude/altitude coordinates into it. The program must compile all user inputs and parameters into a string for transmittal. The program must reject false or nonsense values, and must check to see that all inputs have been given before passing the string to the PEN-X controller.

1.2 Specifications for Each Program

1. C Program – PC

1.) File Name: test.cpp
2.) File Size: 12.2 Kb
3.) Version: Beta 1.1
4.) Date Last Modified: 4/22/04
5.) Description: Allows complete text entry for all target location specs, including hemisphere selection, altitude selection, velocity magnitude/direction selection, acceleration magnitude selection, x-coordinate, y-coordinate, z-coordinate entry. No error detection built in, as this program is just a foundation for future programs.

2. C Program Project – PDA

1.) File Name: location.mcp
2.) File Size: 10.0 Kb
3.) Version: Beta 3.0
4.) Date Last Modified: 4/30/04
5.) Description: Brings up form interface for PDA. Allows selection of buttons to bring up dialog boxes and other forms. These dialog boxes have checkboxes within them for selection
of particular aspect of location or particular range for location (velocity, coordinates, etc.) No text entry allowed.

3. C Program Project – PDA
   1.) File Name: location.mcp
   2.) File Size: 10.0 Kb
   3.) Version: Beta 4.0
   4.) Date Last Modified: 5/2/04
   5.) Description: Brings up form interface for PDA. Allows selection of buttons to bring up dialog boxes and other forms. These dialog boxes have checkboxes within them for selection of particular aspect of location. For range aspects of location, text entry is allowed into particular fields, however, values are not saved due to the limited file size, and the PDA crashes.

2. Bill of Materials

<table>
<thead>
<tr>
<th>Component</th>
<th>Model</th>
<th>Quantity</th>
<th>Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salary</td>
<td>Devan Williams</td>
<td>720 hrs</td>
<td>$25.00/hr</td>
<td>$18,000</td>
</tr>
<tr>
<td>Salary</td>
<td>Toney Jacobson</td>
<td>720 hrs</td>
<td>$25.00/hr</td>
<td>$18,000</td>
</tr>
<tr>
<td>PDA</td>
<td>Palm Tungsten C</td>
<td>2</td>
<td>$400.00</td>
<td>$800.00</td>
</tr>
<tr>
<td>PDA Software</td>
<td>Metrowerks Codewarrior for PalmOS</td>
<td>1</td>
<td>$400.00</td>
<td>$400.00</td>
</tr>
<tr>
<td>IR Receiver</td>
<td>Infrared Receiver</td>
<td>1</td>
<td>$100.00</td>
<td>$100.00</td>
</tr>
<tr>
<td>Printed Circuit</td>
<td>PCBExpress Circuit Board</td>
<td>1</td>
<td>$50.00</td>
<td>$50.00</td>
</tr>
<tr>
<td>Code Book</td>
<td>Palm OS Programming Book</td>
<td>1</td>
<td>$40.00</td>
<td>$40.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>$37,390</strong></td>
</tr>
</tbody>
</table>

3. PEN-X Controller

The PEN-X controller (Figure 2) was developed by a group of Sandia National Laboratories personnel led by Laurence Mayer. The core of the controller is a Linux-based processor, which contains code written by Toney Jacobson and Devan Williams that compares data outputted from a commercial GPS unit and parameters of desired location stored within the software. The processor then creates a unique signal based on this comparison and outputs it to the stronglink. The controller has standard DB9 serial ports for communication between the CPU and an external device (where the interface will connect), a modem (unused for this
project), a stronglink, and a power connector (to connect to the 12 V battery and produce a 5 V signal across two terminals which protrude from the controller).

4. GPS Module

The GPS unit (Figure 3) is the Garmin VI commercial GPS unit, which receives data location from a network of 48 satellites. The unit outputs a text string of location and velocity information to the PEN-X controller with a sampling rate of 1s. This unit connects to the PEN-X controller via a DB9 RS-232 serial port and provides the location information necessary for location enablement.

5. Project Files List

The following files are included in the project to create our application:

AppResources       <DIR>
    “Resources.rcp”
    “Resources.r”

AppHeaders         <DIR>
    “Constants.h”
    “CWDdebugDefines.h”
    “Form.h”
    “ResourceDefines.h”
    “Utils.h”

AppSource          <DIR>
    “Main.c”
    “LocationForm.c”
    “MainForm.c”
    “Utils.c”

6. Selected Project File Code