Collar_DropOff_RevDp0

This is revision D.0 of the Collar Dropoff program. This is the original working program as designed by the Senior Design Group 5, F/S 2004-2005, Boise State University. This program contains the following functionality:

- Gathers both UBLOX proprietary Binary GPS data, and NMEA standard data. All UBLOX data is stored in a single file for retrieval at a later date. Only the most recent NMEA data is stored in a file named NMEA.txt that is intended to be transmitted via the transciever to a remote server. This data can be used for real-time tracking of an animal.
- Utilizes a file named "time.txt" to determine start and stop dates for the collar. The collar hardware will start and find an initial GPS fix before going to sleep until the time indicated in the time.txt file. It then proceeds to gather data at the time interval defined in the time.txt file until it reaches the end of the defined run time (also in the time.txt file).
- Collar is enabled with a remote drop-off device that will activate according to the drop-off time and date in the time.txt file. This is simply a FET switch that activates the device.

This collar hardware is configurable through the Full_Collar_Config program revisions D.?. This program is designed to operate on collar PCB hardware that is Revision D.?. Please do not use with any other revision of the hardware.

Descriptions of source code functionality are as follows:
Collar.h: This is the header file for the entire collar program. This file contains all of the pin definitions, global constants, and prototypes for functions. All subsequent functions include this file as a means of accessing globally defined parameters and ports. This file also includes all of the necessary C-standard libraries as well as CF2-specific libraries provided by Persistor.

Collar.c: This is the main program code. The specification for this code is as follows:

1) Defines all local TUPorts and variables for use in this main code. Unless pointers to different variables are explicitly passed, all subsequent functions define their own variables for local use.

2) Checks the clock speed and outputs the collar code revision data (including the time and date) to the standard output. The standard output is defined as the RS232 IO pins.

3) This code has the capability of defining an Autoexec.bat file on the CF card. This has been omitted, however, as this code is designed to run as the main application (stored in flash and accessed using "boot app" command). This is due to the sleep functions that are utelized in the main program. To create and manage and Autoexec.bat file, uncomment the appropriate code.

4) Calls the "GetTimeInts()" function to read in the contents of the "time.txt" file. This will automatically read in the file's contents and set all timing and start/stop parameters. If the file does not exist, or the file is corrupt in any way, the program will print out an error and exit.

5) Opens the GPS and Radio ports to make them available for the remainder of the program duration.

6) Retreives the initial fix to set the clock and determine all timing parameters.

7) The bulk of the program starts inside the "while" loop that will continue to run the program until it detects that the PBM (reset) button is pressed. While inside the loop, it will retrieve data and transmit the data on the pre-defined intervals contained in the "time.txt" file.

7-a) The program first retrieves the NMEA data. This data is stored in the file named "NMEA.txt" and as local variables for use by the radio when transmitting the NMEA Beacon. This data is over-written with every subsequent NMEA fix so only the most recent fix is contained in this file. This file is not being utelized by the program to create the NMEA Beacon.

7-b) The program then retreives UBLOX proprietary Binary data. This is appended to the file named "UBLOX.ubx". This data is never over-written and is intended to be the bulk GPS data storage.

7-c) If the defined NMEA Beacon interval has been reached, the program will transmit the NMEA beacon based on the local variables that are re-written with every valid NMEA fix. If the NMEA Beacon interval has not been reached, the program will continue without transmitting.

7-d) The program then closes all the ports prior to going to sleep. This is to avoid any errors and to reduce the power consumption while in sleep mode.

7-e) This program is configured to use the PIT (Program Interrupt Timer) method to sleep. This function uses slightly more power than the Suspend method, but has the advantage of fast wake-up times and does not lose data stored in local variables while asleep. In PIT mode, the microcontroller is idle but power is still applied. In Suspend mode, the Microcontroller is powered-down. Revision D of the hardware is not configured to use Suspend mode.

7-f) The program wakes up and opens all the appropriate ports again. It will then loop until power is cut, the PBM (Reset) button is pressed (immediately after the collar wakes), or it reaches the end of the run-time (defined as the stop, or drop-off, time of the collar).

GetTimeInts.c: This is a function that is available to the main code that allows the program to read in all of the parameters defined in the "time.txt" file. This function will read in the contents of the file and store the appropriate parameters in temporary variables. It then calls the appropriate functions to convert those parameters to a date/time structure that is recognizable by the computer. It also reads in the MAC ID (or identifier) of the collar that can be used to identify unique collars by the NMEA beacons that they transmit. When this program is used in conjunction with a "Full_Collar_Config" program (Revision appropriate) the MAC ID of the transciever is read directly from the transceiver and entered in the appropriate location in the "time.txt" file. It is highly recommended that a "Full_Collar_Config" program be used to write the "time.txt" file for this reason. The MAC ID of the transciever must be included in the file to enable the networking capabilities of these collars.

SleepUntilTimeIntEnd.c: This function is called to put the collar to sleep. It contains all the code necessary to shut down certain parts of the collar and wake them up appropriately. It is also capable of making decisions about the sleep mode depending on whether the collar needs to wake up to transmit a NMEA Beacon, whether it needs to gather data, or whether it needs to sleep until a pre-determined start time.

GetFix.c: This function was originally intended to send all manner of commands to the GPS Reciever. It is currently utelized to invoke a GPS solution from a sleeping GPS Receiver. This function then calls "GetGPSMessage" to buffer all the incoming data to be sorted later. This function has error reporting for different types of errors associated with the GPS Reciever. If this function recieves a good GPS message, it uses the "BeaconCreator" to create the NMEA Beacon that is stored in local variables as well as the "NMEA.txt" file. Errors reported include Port error, Max size exceeded, Checksum, and Reset Button Pressed errors. All errors are recorded in the "errorlog" file. This function writes the "NMEABEAC.txt" file.

BufferNMEAData.c: This function provides a buffer for all incoming NMEA Data. It will report if the max size is exceeded or if there is a timeout while waiting for data. It will also report an error if there is a problem with the GPS port.

BufferUBXData.c: This function provides a buffer for all incoming UBLOX data. It will report if the max size is exceeded, if there is a timeout while waiting for data, or if there is an error with the GPS port. This is the analog to the BufferNMEAData. Both functions serve as a temporary holding place for all GPS data so it may be sorted through later.

GetGPSLine.c: This function sorts through the NMEA data and places a line feed character at the end of each NMEA line. This is useful for sorting through the data later on.

GetGPSMessage.c: This function gets an entire set of NMEA Data for use by the NMEA Beacon. It filters through the data that has been stored in the NMEA buffer (and subsequently modified by the GetGPSLine function) and finds the strings that are needed by the user. These are stored in temporary variables and are transmitted via the Transceiver as the NMEA Beacon.

GetUBLOX.c: This function parses through all of the UBLOX GPS data that is contained in the UBLOX buffer and searches for specific strings that will be appended as a fix to the UBLOX.ubx file. The strings that are supported in this revision as follows:

- B5620102: NAV-POSLLH (Geodetic Position Solution)
- B5620104: NAV-DOP (Dilution of Precision)
- B5620103: NAV-STATUS (Receiver Navigation Status - can be used to determine 3D fix)
- B5620106: NAV-SOL (Solution Information - can be used to determine 3D fix)
- B5620121: NAV-TIMEUTC (GPS Time Solution)

- B5620130: NAV-SVINFO (Space Vehicle Information)
- B5620131: NAV-DGPS (Description unavailable)
- B5620210: RXM-RAW (Raw data used for Differential Correction)

If these strings are available in a valid fix, they will be appended to the end of the "UBLOX.ubx" file as the most recent fix.

GetUBLOXfix.c: This function sends the command to the GPS Reciever to wake it up and acquire a UBLOX fix. It then buffers all the incoming data to be sorted and stored by other functions. This function will print errors to the screen regarding the status of the GPS ports, length of the incoming GPS data, etc. This function is what is called from the main program to obtain a UBLOX fix. Please refer to the source code for further information.

Length.c: This is a replacement function for the standard C function length.

Radio.c: This function sends the NMEA Beacon out the radio. The NMEA Beacon is stored in a temporary variable and in the NMEABEAC.TXT file. This function is responsible for turning the radio on and off, monitering the status of the pin In_Range, and checking CTS to see if the transmit buffer has cleared after a transmission. At one point this could print errors to the errorlog file, but that has since been commented out.

RemoteCommand.c: This function was intended for remote commands and is still in the testing phase. It has not been utilized in this version of the program. This file will be removed before release of the final program.

SleepUntilTimeIntEnd.c: This function puts the CF2 to sleep between GPS fixes and NMEA transmissions. It monitors when the CF2 should awaken next, and determines how long it needs to sleep. It awakens and prompts the program for the appropriate action by setting the status of Priority. It then increments the counter that controls intervals between fixes and transmissions. It does so dynamically so every fix is almost exactly the defined interval apart. This function utilizes LPSTOP() due to the hardware configuration.

StripFields.c: This function strips the various strings of interest from the NMEA format data that comes from the GPS Reciever. All fields are defined in the header file.

VerifyChkSum.c: This function verifies the checksum of the incoming NMEA Data. This ensures that the data recorded is valid.

Known issues with this revision of the collar program:
- NMEA Beacon is stored in a variable pointer rather than being read directly from the nmeabeac.txt file. This file was put in place to be utilized when generating the NMEA beacon to tranmsit over the spread-spectrum radio. This results in NMEA data being lost before transmission. If the transmit interval is set to the same value as the GPS fix interval, more often than not, three blank lines and the MAC ID of the collar will be transmitted. This issue is corrected in later revisions of the software.

- This version of the program requires that all start and stop dates (and times) be entered into the time.txt file in UTC time. This results in the user needing to calculate the local time based off of UTC. This issue is corrected in later revisions of the collar program. However, the collar program must be tailored to the time-zone and re-compiled.
- This program calls for a GPS fix twice to obtain a UBLOX and NMEA GPS strings. This has the downside of keeping the GPS receiver running for twice as long as necessary to obtain a fix.

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Collar_UHF_RevDp0

This is revision D.0 of the Collar Dropoff program. This is the original working program as designed by the Senior Design Group 5, F/S 2004-2005, Boise State University. This program contains the following functionality:

- Gathers both UBLOX proprietary Binary GPS data, and NMEA standard data. All UBLOX data is stored in a single file for retrieval at a later date. Only the most recent NMEA data is stored in a file named NMEA.txt that is intended to be transmitted via the transciever to a remote server. This data can be used for real-time tracking of an animal.
- Utilizes a file named "time.txt" to determine start and stop dates for the collar. The collar hardware will start and find an initial GPS fix before going to sleep until the time indicated in the time.txt file. It then proceeds to gather data at the time interval defined in the time.txt file until it reaches the end of the defined run time (also in the time.txt file).
- Collar is enabled with a UHF Transmitter that will transmit while the collar is sleeping and turn off when the collar is awake. This is controlled by the same pin as the Remote Drop Off device.

This collar hardware is configurable through the Full_Collar_Config program revisions D.?. This program is designed to operate on collar PCB hardware that is Revision D.?. Please do not use with any other revision of the hardware.

Descriptions of source code functionality are as follows:
Collar.h: This is the header file for the entire collar program. This file contains all of the pin definitions, global constants, and prototypes for functions. All subsequent functions include this file as a means of accessing globally defined parameters and ports. This file also includes all of the necessary C-standard libraries as well as CF2-specific libraries provided by Persistor.

Collar.c: This is the main program code. The specification for this code is as follows:

1) Defines all local TUPorts and variables for use in this main code. Unless pointers to different variables are explicitly passed, all subsequent functions define their own variables for local use.

2) Checks the clock speed and outputs the collar code revision data (including the time and date) to the standard output. The standard output is defined as the RS232 IO pins.

3) This code has the capability of defining an Autoexec.bat file on the CF card. This has been omitted, however, as this code is designed to run as the main application (stored in flash and accessed using "boot app" command). This is due to the sleep functions that are utelized in the main program. To create and manage and Autoexec.bat file, uncomment the appropriate code.

4) Calls the "GetTimeInts()" function to read in the contents of the "time.txt" file. This will automatically read in the file's contents and set all timing and start/stop parameters. If the file does not exist, or the file is corrupt in any way, the program will print out an error and exit.

5) Opens the GPS and Radio ports to make them available for the remainder of the program duration.

6) Retreives the initial fix to set the clock and determine all timing parameters.

7) The bulk of the program starts inside the "while" loop that will continue to run the program until it detects that the PBM (reset) button is pressed. While inside the loop, it will retrieve data and transmit the data on the pre-defined intervals contained in the "time.txt" file.

7-a) The program first retrieves the NMEA data. This data is stored in the file named "NMEA.txt" and as local variables for use by the radio when transmitting the NMEA Beacon. This data is over-written with every subsequent NMEA fix so only the most recent fix is contained in this file. This file is not being utelized by the program to create the NMEA Beacon.

7-b) The program then retreives UBLOX proprietary Binary data. This is appended to the file named "UBLOX.ubx". This data is never over-written and is intended to be the bulk GPS data storage.

7-c) If the defined NMEA Beacon interval has been reached, the program will transmit the NMEA beacon based on the local variables that are re-written with every valid NMEA fix. If the NMEA Beacon interval has not been reached, the program will continue without transmitting.

7-d) The program then closes all the ports prior to going to sleep. This is to avoid any errors and to reduce the power consumption while in sleep mode.

7-e) This program is configured to use the PIT (Program Interrupt Timer) method to sleep. This function uses slightly more power than the Suspend method, but has the advantage of fast wake-up times and does not lose data stored in local variables while asleep. In PIT mode, the microcontroller is idle but power is still applied. In Suspend mode, the Microcontroller is powered-down. Revision D of the hardware is not configured to use Suspend mode.

7-f) The program wakes up and opens all the appropriate ports again. It will then loop until power is cut, the PBM (Reset) button is pressed (immediately after the collar wakes), or it reaches the end of the run-time (defined as the stop, or drop-off, time of the collar).

GetTimeInts.c: This is a function that is available to the main code that allows the program to read in all of the parameters defined in the "time.txt" file. This function will read in the contents of the file and store the appropriate parameters in temporary variables. It then calls the appropriate functions to convert those parameters to a date/time structure that is recognizable by the computer. It also reads in the MAC ID (or

identifier) of the collar that can be used to identify unique collars by the NMEA beacons that they transmit. When this program is used in conjunction with a "Full_Collar_Config" program (Revision appropriate) the MAC ID of the transciever is read directly from the transceiver and entered in the appropriate location in the "time.txt" file. It is highly recommended that a "Full_Collar_Config" program be used to write the "time.txt" file for this reason. The MAC ID of the transciever must be included in the file to enable the networking capabilities of these collars.

SleepUntilTimeIntEnd.c: This function is called to put the collar to sleep. It contains all the code necessary to shut down certain parts of the collar and wake them up appropriately. It is also capable of making decisions about the sleep mode depending on whether the collar needs to wake up to transmit a NMEA Beacon, whether it needs to gather data, or whether it needs to sleep until a pre-determined start time.

GetFix.c: This function was originally intended to send all manner of commands to the GPS Reciever. It is currently utelized to invoke a GPS solution from a sleeping GPS Receiver. This function then calls "GetGPSMessage" to buffer all the incoming data to be sorted later. This function has error reporting for different types of errors associated with the GPS Reciever. If this function recieves a good GPS message, it uses the "BeaconCreator" to create the NMEA Beacon that is stored in local variables as well as the "NMEA.txt" file. Errors reported include Port error, Max size exceeded, Checksum, and Reset Button Pressed errors. All errors are recorded in the "errorlog" file. This function writes the "NMEABEAC.txt" file.

BufferNMEAData.c: This function provides a buffer for all incoming NMEA Data. It will report if the max size is exceeded or if there is a timeout while waiting for data. It will also report an error if there is a problem with the GPS port.

BufferUBXData.c: This function provides a buffer for all incoming UBLOX data. It will report if the max size is exceeded, if there is a timeout while waiting for data, or if there is an error with the GPS port. This is the analog to the BufferNMEAData. Both functions serve as a temporary holding place for all GPS data so it may be sorted through later.

GetGPSLine.c: This function sorts through the NMEA data and places a line feed character at the end of each NMEA line. This is useful for sorting through the data later on.

GetGPSMessage.c: This function gets an entire set of NMEA Data for use by the NMEA Beacon. It filters through the data that has been stored in the NMEA buffer (and subsequently modified by the GetGPSLine function) and finds the strings that are needed by the user. These are stored in temporary variables and are transmitted via the Transceiver as the NMEA Beacon.

GetUBLOX.c: This function parses through all of the UBLOX GPS data that is contained in the UBLOX buffer and searches for specific strings that will be appended as a fix to the UBLOX.ubx file. The strings that are supported in this revision as follows:

- B5620102: NAV-POSLLH (Geodetic Position Solution)
- B5620104: NAV-DOP (Dilution of Precision)
- B5620103: NAV-STATUS (Receiver Navigation Status - can be used to determine 3D fix)
- B5620106: NAV-SOL (Solution Information - can be used to determine 3D fix)
- B5620121: NAV-TIMEUTC (GPS Time Solution)
- B5620130: NAV-SVINFO (Space Vehicle Information)
- B5620131: NAV-DGPS (Description unavailable)
- B5620210: RXM-RAW (Raw data used for Differential Correction)

If these strings are available in a valid fix, they will be appended to the end of the "UBLOX.ubx" file as the most recent fix.

GetUBLOXfix.c: This function sends the command to the GPS Reciever to wake it up and acquire a UBLOX fix. It then buffers all the incoming data to be sorted and stored by other functions. This function will print errors to the screen regarding the status of the GPS ports, length of the incoming GPS data, etc. This function is what is called from the main program to obtain a UBLOX fix. Please refer to the source code for further information.

Length.c: This is a replacement function for the standard C function length.

Radio.c: This function sends the NMEA Beacon out the radio. The NMEA Beacon is stored in a temporary variable and in the NMEABEAC.TXT file. This function is responsible for turning the radio on and off, monitering the status of the pin In_Range, and checking CTS to see if the transmit buffer has cleared after a transmission. At one point this could print errors to the errorlog file, but that has since been commented out.

RemoteCommand.c: This function was intended for remote commands and is still in the testing phase. It has not been utilized in this version of the program. This file will be removed before release of the final program.

SleepUntilTimeIntEnd.c: This function puts the CF2 to sleep between GPS fixes and NMEA transmissions. It monitors when the CF2 should awaken next, and determines how long it needs to sleep. It awakens and prompts the program for the appropriate action by setting the status of Priority. It then increments the counter that controls intervals between fixes and transmissions. It does so dynamically so every fix is almost exactly the defined interval apart. This function utilizes LPSTOP() due to the hardware configuration.

StripFields.c: This function strips the various strings of interest from the NMEA format data that comes from the GPS Reciever. All fields are defined in the header file.

VerifyChkSum.c: This function verifies the checksum of the incoming NMEA Data. This ensures that the data recorded is valid.

Known issues with this revision of the collar program:
- NMEA Beacon is stored in a variable pointer rather than being read directly from the nmeabeac.txt file. This file was put in place to be utilized when generating the NMEA beacon to tranmsit over the spread-spectrum radio. This results in NMEA data being lost before transmission. If the transmit interval is set to the same value as the GPS fix interval, more often than not, three blank lines and the MAC ID of the collar will be transmitted. This issue is corrected in later revisions of the software.
- This version of the program requires that all start and stop dates (and times) be entered into the time.txt file in UTC time. This results in the user needing to calculate the local time based off of UTC. This issue is corrected in later revisions of the collar program. However, the collar program must be tailored to the time-zone and re-compiled.
- This program calls for a GPS fix twice to obtain a UBLOX and NMEA GPS strings. This has the downside of keeping the GPS receiver running for twice as long as necessary to obtain a fix.

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!! This Revision has not been released or extensively tested!!!
February 21, 2006 - Mark Kniep

Collar_DropOff_RevDp1

This is revision D.1 of the Collar Dropoff program. This is the updated version of the original program written by Senior Design Group 5, F/S 2005, Boise State University. This implements improvements meant to reduce errors and enhancements to support remote-data download and collar networking. This program contains the following functionality:

- Gathers both UBLOX proprietary Binary GPS data, and NMEA standard data. All UBLOX data is stored in a single file for retrieval at a later date. Only the most recent NMEA data is stored in a file named NMEA.txt that is intended to be transmitted via the transciever to a remote server. This data can be used for real-time tracking of an animal.
- Utilizes a file named "time.txt" to determine start and stop dates for the collar. The collar hardware will start and find an initial GPS fix before going to sleep until the time indicated in the time.txt file. It then proceeds to gather data at the time interval defined in the time.txt file until it reaches the end of the defined run time (also in the time.txt file).
- Collar is enabled with a remote drop-off device that will activate according to the drop-off time and date in the time.txt file. This is simply a FET switch that activates the device.
- Archives data every Wednesday at the same time as the collar was started. This same functionality will be used in the future to archive data after all remote-data downloads.

This collar hardware is configurable through the Full_Collar_Config program revisions D.?. This program is designed to operate on collar PCB hardware that is Revision D.?. Please do not use with any other revision of the hardware.

Descriptions of source code functionality are as follows:
Collar.h: This is the header file for the entire collar program. This file contains all of the pin definitions, global constants, and prototypes for functions. All subsequent functions include this file as a means of accessing globally defined parameters and ports. This file also includes all of the necessary C-standard libraries as well as CF2-specific libraries provided by Persistor.

Collar.c: This is the main program code. The specification for this code is as follows:

1) Defines all local TUPorts and variables for use in this main code. Unless pointers to different variables are explicitly passed, all subsequent functions define their own variables for local use.

2) Checks the clock speed and outputs the collar code revision data (including the time and date) to the standard output. The standard output is defined as the RS232 IO pins.

3) This code has the capability of defining an Autoexec.bat file on the CF card. This has been omitted, however, as this code is designed to run as the main application (stored in flash and accessed using "boot app", or "app" commands). This is due to the sleep functions that are utilized in the main program. To create and manage and Autoexec.bat file, uncomment the appropriate code.

4) Calls the "GetTimeInts()" function to read in the contents of the "time.txt" file. This will automatically read in the file's contents and set all timing and start/stop parameters. If the file does not exist, or the file is corrupt in any way, the program will print out an error and exit.

5) Opens the GPS and Radio ports to make them available for the remainder of the program duration.

6) Retreives the initial fix to set the clock and determine all timing parameters.

7) The bulk of the program starts inside the "while" loop that will continue to run the program until it detects that the PBM (reset) button is pressed. While inside the loop, it will retrieve data and transmit the data on the pre-defined intervals contained in the "time.txt" file.

7-a) The program first retrieves the NMEA data. This data is stored in the file named "NMEA.txt" and as local variables for use by the radio when transmitting the NMEA Beacon. This data is over-written with every subsequent NMEA fix so only the most recent fix is contained in this file. This file is not being utelized by the program to create the NMEA Beacon.

7-b) The program then retreives UBLOX proprietary Binary data. This is appended to the file named "UBLOX.ubx". This data is never over-written and is intended to be the bulk GPS data storage.

7-c) If the defined NMEA Beacon interval has been reached, the program will transmit the NMEA beacon based on the contents of the "NMEABEAC.TXT" file. This revision of the program retrieves the latest valid NMEA fix from this file and builds the

appropriate NMEA Beacon for transmission. If the NMEA Beacon interval has not been reached, the program will continue without transmitting.

7-d) The program then closes all the ports prior to going to sleep. This is to avoid any errors and to reduce the power consumption while in sleep mode.

7-e) This program is configured to use the PIT (Program Interrupt Timer) method to sleep. This function uses slightly more power than the Suspend method, but has the advantage of fast wake-up times and does not lose data stored in local variables while asleep. In PIT mode, the microcontroller is idle but power is still applied. In Suspend mode, the Microcontroller is powered-down. Revision D of the hardware is not configured to use Suspend mode.

7-f) The program wakes up and opens all the appropriate ports again. It will then loop until power is cut, the PBM (Reset) button is pressed (immediately after the collar wakes), or it reaches the end of the run-time (defined as the stop, or drop-off, time of the collar).

7-g) This program will check whether the appropriate Archive time has been reached. This is done by accessing the function named "scheduled_archive()". If the time interval has been reached, the file named "UBLOX.UBX" will be renamed according to the date. The nomenclature of the archived name is as follows:

A08FEB06.UBX

- The first character is a placeholder that will be incremented if the file is archived more than once a day. For instance, if for some reason the file archived again, the new archive file would be named "B08FEB06.UBX".

- The second and third character indicate the date (day of the month) on which the file was archived.

- The fourth through sixth characters indicate the month that the file was archived.

- The last two characters before the extension indicate the last two digits of the year in which the file was archived.

The name of this file indicates it is the first file to be archived on February 8, 2006.

GetTimeInts.c: This is a function that is available to the main code that allows the program to read in all of the parameters defined in the "time.txt" file. This function will read in the contents of the file and store the appropriate parameters in temporary variables. It then calls the appropriate functions to convert those parameters to a date/time structure that is recognizable by the computer. It also reads in the MAC ID (or identifier) of the collar that can be used to identify unique collars by the NMEA beacons that they transmit. When this program is used in conjunction with a "Full_Collar_Config" program (Revision appropriate) the MAC ID of the transciever is read directly from the transceiver and entered in the appropriate location in the "time.txt" file. It is highly recommended that a "Full_Collar_Config" program be used to write the "time.txt" file for this reason. The MAC ID of the transciever must be included in the file to enable the networking capabilities of these collars.

- This file has been modified since the last program revision. The function will now take local time into account when calculating the program start and stop times. Previously, all times and dates entered into the "TIME.TXT" file were required to be in

GMT (Greenwich Mean UTC Time). With this revision of the program, times may be entered in the local time zone. This revision is released as USA Mountain Standard Time. Simple alterations to the source code (and subsequent re-compilation) can make taylor this program to any time zone. Please refer to the source code for more information.

SleepUntilTimeIntEnd.c: This function is called to put the collar to sleep. It contains all the code necessary to shut down certain parts of the collar and wake them up appropriately. It is also capable of making decisions about the sleep mode depending on whether the collar needs to wake up to transmit a NMEA Beacon, whether it needs to gather data, or whether it needs to sleep until a pre-determined start time.

GetFix.c: This function was originally intended to send all manner of commands to the GPS Reciever. It is currently utilized to invoke a GPS solution from a sleeping GPS Receiver. This function then calls "GetGPSMessage" to buffer all the incoming data to be sorted later. This function has error reporting for different types of errors associated with the GPS Reciever. If this function recieves a good GPS message, it uses the "BeaconCreator" to create the NMEA Beacon that is stored in local variables as well as the "NMEA.txt" file. Errors reported include Port error, Max size exceeded, Checksum, and Reset Button Pressed errors. All errors are recorded in the "errorlog.txt" file. This function writes the "NMEABEAC.txt" file.

BufferNMEAData.c: This function provides a buffer for all incoming NMEA Data. It will report if the max size is exceeded or if there is a timeout while waiting for data. It will also report an error if there is a problem with the GPS port.

BufferUBXData.c: This function provides a buffer for all incoming UBLOX data. It will report if the max size is exceeded, if there is a timeout while waiting for data, or if there is an error with the GPS port. This is the analog to the BufferNMEAData. Both functions serve as a temporary holding place for all GPS data so it may be sorted through later.

GetGPSLine.c: This function sorts through the NMEA data and places a line feed character at the end of each NMEA line. This is useful for sorting through the data later on.

GetGPSMessage.c: This function gets an entire set of NMEA Data for use by the NMEA Beacon. It filters through the data that has been stored in the NMEA buffer (and subsequently modified by the GetGPSLine function) and finds the strings that are needed by the user. These are stored in temporary variables and are transmitted via the Transceiver as the NMEA Beacon.

GetUBLOX.c: This function parses through all of the UBLOX GPS data that is contained in the UBLOX buffer and searches for specific strings that will be appended as a fix to the UBLOX.ubx file. The strings that are supported in this revision as follows:

- B5620102: NAV-POSLLH (Geodetic Position Solution)

- B5620104: NAV-DOP (Dilution of Precision)
- B5620103: NAV-STATUS (Receiver Navigation Status - can be used to determine 3D fix)
- B5620106: NAV-SOL (Solution Information - can be used to determine 3D fix)
- B5620121: NAV-TIMEUTC (GPS Time Solution)
- B5620130: NAV-SVINFO (Space Vehicle Information)
- B5620131: NAV-DGPS (Description unavailable)
- B5620210: RXM-RAW (Raw data used for Differential Correction)

If these strings are available in a valid fix, they will be appended to the end of the "UBLOX.ubx" file as the most recent fix.

GetUBLOXfix.c: This function sends the command to the GPS Reciever to wake it up and acquire a UBLOX fix. It then buffers all the incoming data to be sorted and stored by other functions. This function will print errors to the screen regarding the status of the GPS ports, length of the incoming GPS data, etc. This function is what is called from the main program to obtain a UBLOX fix. Please refer to the source code for further information.

Length.c: This is a replacement function for the standard C function length.

Radio.c: This function has been completely re-written since the last program revision. This function controls all transmissions out of the transceiver. It is responsible for checking the status of the pin "In_Range". Depending on the range status of a server, it will transmit the contents of the memory named "RadioMessage". This may contain a current GPS NMEA fix, or the contents of the "UBLOX.UBX" file (during remote-download, not available in this release). It also monitors the status of the CTS pin of the transceiver. This will indicate whether the transmit buffer has emptied. It will return an appropriate value to the calling function to indicate the result of the transmission. Please refer to the source code for further information on the values returned by this function. This function will transmit data four lines at a time for any transmission. All error handling is built into the transceivers themselves, and is not indicated in the code.

SleepUntilTimeIntEnd.c: This function puts the CF2 to sleep between GPS fixes and NMEA transmissions. It monitors when the CF2 should awaken next, and determines how long it needs to sleep. It awakens and prompts the program for the appropriate action by setting the status of Priority. It then increments the counter that controls intervals between fixes and transmissions. It does so dynamically so every fix is almost exactly the defined interval apart. This function utilizes LPSTOP() due to the hardware configuration.

StripFields.c: This function strips the various strings of interest from the NMEA format data that comes from the GPS Reciever. All fields are defined in the header file.

VerifyChkSum.c: This function verifies the checksum of the incoming NMEA Data. This ensures that the data recorded is valid.

Known issues with this revision of the collar program:
- This program calls for a GPS fix twice to obtain a UBLOX and NMEA GPS strings. This has the downside of keeping the GPS receiver running for twice as long as necessary to obtain a fix.