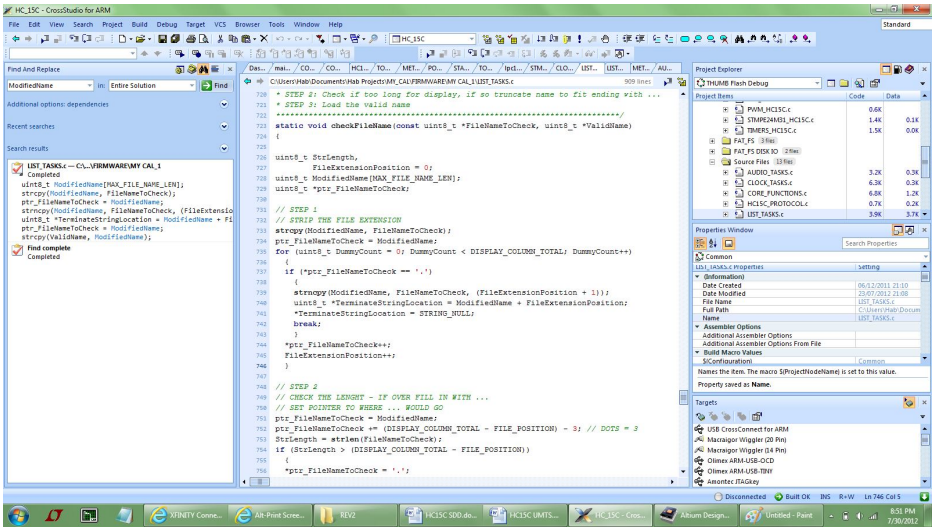
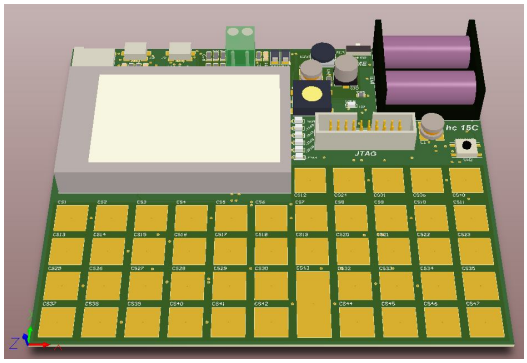
HC15C DEVICE

SOFTWARE DESCRIPTION DOCUMENT





Author: Hab S. Collector
Firmware Enginner: Hab S. Collector
Hardware Engineer: Hab S. Collector
Software Engineer: Hab S. Collector

Table of Contents

DOCUMENT REVISION HISTORY		
DATE	**DESCRIPTION**	**EDIT BY**
15-Jul-12	Initial draft	HSC
31-Jul-12	Initial release	HSC

1. INTRODUCTION:

This is the Software Description Document for Hab's HC15C firmware. As this is a firmware document it should be reviewed with the HC15C schematic. For clarity, and where necessary the HC15C Windows Application Software (Win App) is also covered. The intent of this document is not a line by line, or function by function review of the software. The code in of itself is well commented and constructed with the intent that it can be reviewed by an experienced firmware engineer. The schematic is also designed such that it can be reviewed by an experience hardware engineer. This document, in very broad terms, covers how the code is constructed, and the interconnections of the associated tasks.

2. INTENDED AUDIENCE AND PRE-REQUISTS TO READING:

The intended audience is hardware and firmware engineers seeking an in-depth knowledge of the HC15C device operation. The prerequisite to this document is the HC15C User Manual and Technical Specifications (UMTS) document. Said documents form the fundamental core of what you need to understand before reading this document. Reading this document before the UMTS may result in wasted time. This document should be read in its entirety from start to end. The information presented here builds upon itself rather than repeats.

3. THE BIG PICTURE:

The HC15C target processor is the NXP LPC1768 microcontroller clocked from an external 12MHz crystal. The core clock runs at 100MHz. Note the device can also run on an LPC1769 with a core clock speed of 120MHz. The HC15C firmware is built upon the embedded Rowley CTL (Crossworks Tasking Library) RTOS version 2.1. The CTL is a priority based, preemptive, RTOS scheduler. Its implementation is similar in construct to Free RTOS, but with additional capabilities. The CTL works within the Rowley ARM Crossworks Integrated Development Editor (IDE VER 2.1.1.2011). The project is the HC_15C and it is the only project within the solution of the same name. All task files can be found under SOURCE directory. The lower level supporting drivers can be found in the DRIVER directory. It is noted at this time, that the LPC17xx has an M3 core, hence the driver level functions take full advantage of the CMSIS libraries loaded to the CMSIS directory. Chan's FAT File System's port is loaded to FAT_FS and FAT_FS DISK IO directories. The USB Virtual COM port is loaded to the VCOM USB directory.

The HC15C Windows Application (Win App) Software is built on Microsoft Visual C++ compiler, in the visual studio environment. Win App software components use the .NET architecture. The Win App is intended for use on the Windows 7 Operating System, though it should work equally well on Windows XP. The project was complied in MS Visual Studio 2010.

4. HC15C MODES AND TASKS:

The HC15C device has many unique built in capabilities. These capabilities: Calculator, Clock, Volt Meter, Ohm Meter, Music, Memory Stick Reader, etc. etc. can be thought of as different MODES of operation. The modes of operation are described as:

1. Calculator Mode
2. Setup Mode
3. Clock Mode
4. Alarm Mode
5. Volt Meter Mode
6. Ohm Meter Mode
7. SD List Mode
8. Music Mode

It is important to understand that within each mode of operation the device behaves differently. That within these unique modes, different tasks are made runnable, and along with those tasks, different IRQs and hardware peripherals are enabled. Hence the starting of a mode begins with the starting of that mode's startup function. The following functions start the given modes:

1. call_CalMode
2. call_ClockMode
3. call_AlarmMode
4. call_SetupMode
5. call_SD_ListMode
6. call_MusicListMode
7. call_VoltMeterMode
8. call_OhmMeterMode

As there are tasks and interrupts unique to each mode, these tasks and interrupts must be likewise terminated before a new mode can begin. The function call_EndPresentMode terminates the present mode before the next mode can begin. Modes are terminated using the functions:

1. call_CalModeEnd
2. call_ClockModeEnd
3. call_AlarmModeEnd
4. call_SetupModeEnd
5. call_SD_ListModeEnd
6. call_MusicListModeEnd
7. call_VoltMeterModeEnd
8. call_OhmMeterModeEnd

Beyond the modes of operation the tasks associated with each mode helps to build a more complete understanding of how the firmware works. As the firmware is built upon an embedded RTOS, the key to its understanding lies within the tasks, and interconnections of the tasks. The project is built upon eight tasks, plus the idle task. As CTL does not specifically define an idle task, the main loop becomes the idle task, running at the lowest system priority after all other tasks go idle. The tasks associated with the project in order of priority are:

1. The System Init task "init_task"

2. The Power task “batQ_task”
3. The Audio task “audio_task”
4. The Setup task “setup_task”
5. The Clock task “clock_task”
6. The Meter task “meter_task”
7. The List task “SD_List_task”
8. The Click task “click_task”

5. THE CAL SETTINGS STRUCTURE:

The overall governing data structure of the HC15C device is the CalSetting Structure. This is the most ubiquitous variable found throughout the project. The structure is of type Type_HC15C which is describe here:

```
typedef struct
  {
  uint8_t CalError;         // CAL ERROR SEE ENUM ATTENTION CODE
  uint8_t CalBase;          // CAL BASE SEE ENUM NUMERIC BASE
  uint8_t FixPrecision;     // FIX FORMAT PRECISION FOR BASE 10
  uint8_t EngPrecision;     // ENG FORMAT PRECISIION FOR BASE 10
  uint8_t DisplayMode;      // CAL DISPLAY SEE ENUM DISPLAY MODE
  uint8_t CalAngle;         // CAL ANGLE SEE ENUM ANGLE MEASURE
  uint8_t CalMode;          // CAL MODE SEE ENUM OPERATING_MODE
  uint16_t CalVerboseMask;  // CAL MODE FULL INTERACTIVE HELP
  BOOLEAN L_Shift;          // LEFT SHFIT FLAG
  BOOLEAN R_Shift;          // RIGHT SHIFT FLAG
  BOOLEAN USB_Link;         // USB VCOM OK TO USE
  BOOLEAN SleepDisable;     // USED TO DISABLE SLEEP
  uint32_t Mask_KeyTouchA;  // TOUCH A KEYPAD MASKED KEYS
  uint32_t Mask_KeyTouchB;  // TOUCH A KEYPAD MASKED KEYS
  Type_Setup Setup;         // CALCULATOR GENERAL SETTINGS
  Type_TimeAlarm TimeAlarm; // TIME ALARM STATUS AND EVENTS
  Type_MusicPlayBack MusicPlayBack;     // MUSIC PLAY
  } Type_CalSettings;
```

The member structures are defined here…

```
typedef struct
  {
  uint8_t BackLightTimeOut;
  uint8_t CalVerbose;
  uint8_t TimeToSleep;
  } Type_Setup;

typedef struct
  {
```

```
  BOOLEAN AlarmEnable;
  BOOLEAN AlarmEvent;
  BOOLEAN HC15C_InDeepSleep;
  } Type_TimeAlarm;

typedef struct
  {
  BOOLEAN Play;
  BOOLEAN Playing;
  BOOLEAN Pause;
  volatile uint32_t PlayTimeInSeconds;
  } Type_MusicPlayBack;
```

This data structure is key to the device operation as it contains information necessary to the operational and mode specific parameters of the device.

By understanding the modes of operation you understand how the firmware works. We will examine each mode, but before doing so we will review two unique tasks: the click and startup tasks.

6. "THE" CLICK TASK:

The most fundamental and hence highest priority task is the click task. The click task is the UI via 48 Capacitive Touch Sensors (keys) operated across two devices (see schematic STMPE24M31 A and B). From a big picture perspective the touching of a key generates one of two high priority IRQs (INT_ACAP and INT_BCAP) – These devices are read via an I2C interface. The A keys are keys located on the left side of the device. From left to right they encompass the rows headed by the alpha-hexadecimal keys. The B keys are all keys in rows to the right of alpha-hexadecimal digits. The corresponding IRQ is processed and loaded to a message queue which is serviced by the click task. As different modes of operation enable / disable different keys, one of the click task duties is to filter key press events. If a corresponding key is not masked by the present filter, then the corresponding function is called. It is the responsibility of the various mode functions to mask keys unique to their mode of operation.

7. START UP TASK:

The start task is responsible for:

- Power On Reset (POR) system startup activities
- Wake from power conservation state (Deep Sleep)

On POR the startup task is called to initialize all hardware peripherals such as the SPI bus LCD interface, the I2C Capacitive Touch Interface, etc, etc. and place the HC15C in calculator mode (the most used mode). The task also loads from non-volatile EEPROM numerous operational parameters. These parameters include:

1. The four deep stack value and display mode
2. Angular measure (radian or degree)
3. Notation display (decimal or hexadecimal)
4. The Setup Parameters (see Setup Mode)
5. The memory locations (0-99): Note a smart save algorithm is employed to conserve EEPROM storage.

The wake from deep sleep is similar to the POR. Wake from deep sleep occurs if the user touches a key (Touch A or B external IRQ EINT2 and EINT3 respectively), or presses the WAKE button (external IRQ EINT0). In wake from deep sleep only the hardware peripherals that were suspended by sleeping are initialized. Wake from deep sleep can also occur as a result of the RTC alarm interrupt. The startup task is only runnable on POR or wake from deep sleep. As such it is the lowest priority task.

It is noted here that every time the idle task runs (there are no other active tasks) that the device will be placed in a state of light sleep to conserve power. The idle task sleep is not a deep sleep, as such, it wakes from every device IRQ.

8. POWER TASK:

The power task is responsible for:

- Battery life monitor and update
- Backlight output level
- No activity power down (sleep) events
- Connection of USB device (detect USB power)

A one second timer IRQ is used to set the power task event which monitors battery life. The task uses the ADC peripheral to measure battery voltage. The voltage is compared against a lookup table of values and the battery life display is updated upon change of status only. This same event measures the USB voltage to determine if a connection has been made. If so it enables various settings, making the App Link mode possible.

Similarly a timer IRQ sets an event that signals the task to set the backlight dimming level. The backlight is controlled via a PWM positive duty cycle peripheral. The task monitors a countdown variable and at computed intervals of no UI lowers the output to the backlight.

As these events are not time critical to the user experience the power tasks is the next lowest priority task.

It is noted here that the HC15C device has a “hardware only, automatic detect and switch to USB power circuit”. In the presence of USB power the device will run from the computer’s power and not its batteries. If USB power is removed the circuit switches back to battery operation. This switch is performed without interrupting the user.

9. CALCULATOR MODE:

This is the default mode of operation for the HC15C device. Most interestingly this mode is only associated with the click task. Touching a key, loads a key press message to a queue. Touch messages are extracted from the queue in FIFO fashion. The message is compared against a filter, if unmasked (meaning the key press is valid), the button click sound is loaded to the audio queue for playing and the function or input associated with that key is processed. Note as the click task is the highest priority task, the calculator function or input is processed first. Though the audio queue is runnable, the click is played only when the audio task's priority allows it to execute. In normal operation the order of events is indistinguishable to the user. However if you quickly piano role the touch keys for fun, the display will update and compute ahead of the "click". The back log of clicks will all play as the click task allows. It is noted here that all of the math related functions can be found in the files MATH_FUNCTIONS.C and .H. Many of the core, under lining functions that support the RPN method will be found in CORE_FUNCTIONS.C and .H.

One of the few drawbacks of the C language is that it does not support error trapping. As such, much of the math library is built around the prediction of mathematical or unsupported modes. The device does not directly support infinite results or imaginary numbers.

10. SETUP MODE:

Setup mode allows the user to set:

- Time to Back Light Off
- Cal Verbose Level
- Time to deep sleep

Setup mode works out of the setup task. All three parameters are environmental parameters of the device and are stored to non-volatile EEPROM via an I2C bus interface. It should be noted that changes to a parameter automatically causes it to be the active value. These parameters, as all stored operating conditions, are saved as a result of time out to deep sleep, or forced sleep. The list of stored parameters can be viewed in the driver file CAT24C2.C and .H

11. CLOCK MODE:

Clock mode displays the present time and date along with a stopwatch feature. Clock mode operates from the clock task. Upon start clock mode loads the battery back RTC Time and Date. A stopwatch is also displayed (set to 0). Clock mode enables the RTC for a one second interrupt which sets an event causing the clock task to run. The clock task updates the change in time at the second, minute hour and date only as necessary. When the user starts the stop watch, a 100ms timer event is enabled. The timer sets an event causing the task to increment the stop watch. The stop watch is incremented in hundreds of milliseconds, seconds, minutes and hours only as necessary. The stopwatch can be paused (timer disabled) and reset. It is important to note that a running stopwatch will disable deep sleep.

Clock Mode works with the Win App if the device USB (J3 only) is connected to a computer. If the USB is connected the App Link key is enabled. The clock's date and time can only be set by the Win App. When connected activate the Win App by touching the App Link key and launching the Win App software (not necessarily in that order). From the Win App Software connect to the HC15C. The time set button on the Win App software is enabled by the download key on the HC15C device. The HC15C communicates to the Win App via a custom protocol. This protocol can be reviewed in the files HC15C_PROTOTOL.C AND .H

12. ALARM MODE:

The purpose of alarm mode is to set an alarm. Alarm mode is a subset of the clock task. The left and right keys allow navigation to set the hour, minute and day / night (AM/PM) time. Setting the alarm enables the RTC alarm interrupt. The RTC alarm interrupt is masked for hour and minute only. In other words an alarm event is set only upon match of hour and minute. The hour time is military time so it encapsulates AM / PM. Upon alarm, the IRQ closes the present mode and presents the time to the user. Stopping the alarm disables the RTC alarm interrupt.

13. VOLT METER MODE:

Built into the device is a highly accurate volt meter. Users should consult the UMTS document for specifications and operating parameters. The volt meter mode is a subset of the meter task. The mode starts by turning on / off the control logic (FETs in the meter circuit) necessary to measure voltage. A timer interrupts sets an event which causes the ADC to measure the input voltage on J5. An algorithm within the task acts as a smart filter to accept or reject the measured value. If rejected, the rejected value is stored for future comparison to determine if it was in fact valid. Note this is not a simple averaging filter. The averaging filter takes place in the ADC measure function ADC_getConvertedValue (ADC_HC15C.C and .H) which incorporates a filter average and sample interval. The measure value also works to establish the input voltage divider ratio. This divider allows for the maximum dynamic range of the 10bit ADC. The input divider ratio is analogues to the voltage range of operation. The range is set by turning on one of three FETs that control the divider ratio. Additionally a max value read is maintained and update as necessary as part of the measure. This value can be reset by touching the reset key.

Volt meter mode works with the Win App device. When connected to the Win App (Volt Meter Tab) the HC15C device transmit the measured voltage via custom protocol. The raw packet data is displayed to the Win App.

14. OHM METER MODE:

Ohm meter mode is somewhat miss-labeled as it would be better called continuity mode. The ohm meter is a subset of the meter task. The mode operates in very similar fashion to volt meter mode. Ohm meter mode allows the user to increment the threshold for continuity. This threshold is changed in increments of 25 ohms. When continuity is reached (the measured

resistance is less than or equal to the set resistance) the task enables a timer IRQ. The timer IRQ alternates the audio level to the DAC output between two states (high and low) to produce a 2KHz audio tone. Ohm meter mode works with the Win App in similar fashion to the volt meter.

15. SD LIST MODE:

This mode allows the user to view the contents of the removable SD card. SD Mode is a subset of the list task. The list task, as well as all other functions that access the SD card, use Chan's embedded FAT File System. The disk IO layer of FAT FS is via the SPI bus interface to the SD card. The list shows both file names and directories.

16. MUSIC LIST MODE:

Music List mode allows the user to play music files stored to the directory MUSIC_FILES on the SD card. Music List mode is a subset of the list task. Users can scroll the list, select a song: Play, Pause, and Stop. Audio is played by posting the selected audio specific structure to the audio task message queue. The audio task uses a proprietary 16 bit PCM file player function call_play16Bit_WAVE. This player is encapsulated within the function AUDIO_TASKS.C and .H. Generally speaking the function works by loading a defined block of data. That data is separated to left and right channels (left and right (if stereo)) and scaled for a 10 bit DAC. The data is then placed to left and right circular buffers. A timer IRQ which has been initialed to interrupt at the audio playback rate is used to extract the file's audio from the circular buffer and load the DAC peripheral. A peak detector algorithm sets the dB marking LEDs. It should be noted that all audio is played by the audio task. Playing the audio from music mode also enables the one second RTC interrupt. This interrupt is used to decrement the play time as the file is played.