M851 Counter WristApp Design

Timex Corporation
July 2002

DOCUMENT REVISION HISTORY

REVISION: 1.0	DATE: 07/31/2002	AUTHOR: NINO ALDRIN L. SARMIENTO

AFFECTED PAGES	DESCRIPTION
All	Created document.

TABLE OF CONTENTS

1 Introduction

The M851 Kernel is a platform that is geared for developing a variety of applications that can be incorporated into the operating system during power up or downloaded to EEPROM through USB Datalink communications. Refer to the M851 Application Design Guide for an overview of the M851 Kernel and how applications are processed in the M851 Kernel.

This document serves as a guide for developing a WristApp.

1.1 Applicable Documents

The following documents serves as detailed reference in the creation of this document.

- M851 Application Design Guide
- M851 WristApp API Reference Guide
- S1C88349 Core CPU Manual

2 COUNTER WristApp: Putting it all together.

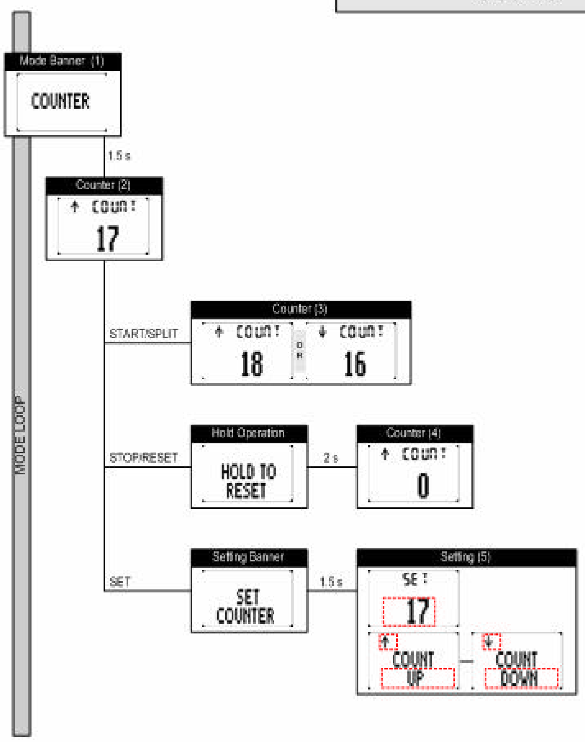
This section will go through the process of building a wristapp – the Counter WristApp – from design, compile and downloading the application to the watch. This application is simple and does not require any database access.

⚠ **WARNING**: *There is no debugging capability once the WristApp is downloaded into the watch. You will either have a fully operational wristapp or the watch resets during WristApp execution.*

2.1 Specification

The diagram below shows how the counter wristapp operates and how it interacts with the user inputs.

Counter



NOTES

1. This display appears only if this mode is enabled. Otherwise, the next mode banner shall be displayed.

2. The counter's current value is displayed in large digits in the main dot matrix.

   The counter's current direction is indicated in the upper dot matrix, with an up arrow if UP is selected, or a down arrow if DOWN is selected.

3. The counter's value is incremented or decremented depending upon the direction setting described below. If the counter value is at its minimum or maximum, it cannot be decremented or incremented, respectively; the button shall have no fuction.

   The Button Beep (see Appendix D) shall be generated for each press of START/SPLIT.

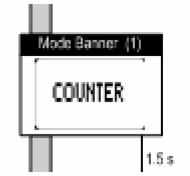
4. The counter is set to zero. The counter direction is unchanged.

5. In the first step, the user may set the current counter value. In the second step, the user may set the counter direction.

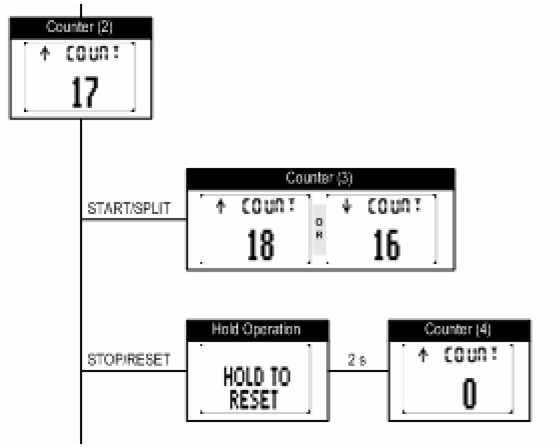
2.2 States

2.2.1 State Transition Diagram

The specification can be broken down into its basic components. The counter application can be grouped into 4 distinct operations: banner, default, set banner and set operations.

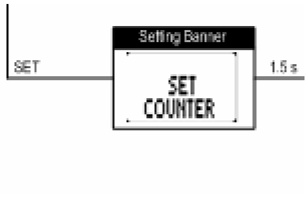


The Banner State Handler. This involves mainly displaying the name of the mode. We need to design this handler to allow the M851 PIM to display the user specified mode banner. Notice the required 1.5 second timeout prior to going into default mode.

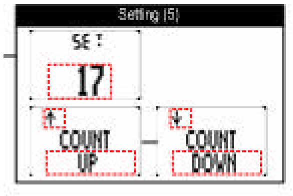


The Default State Handler. This is the main interface of the application.

NOTE: The hold-to-reset operation may be put into a different state handler to simplify the number of events the default state handler will process. Since this is a small application, putting the reset operation inside the default state handler is easily facilitated.



The Set State Banner Handler. By convention, this is a required state prior to going to the actual setting state. Notice the required 1.5 second timeout prior to going into the set state handler.

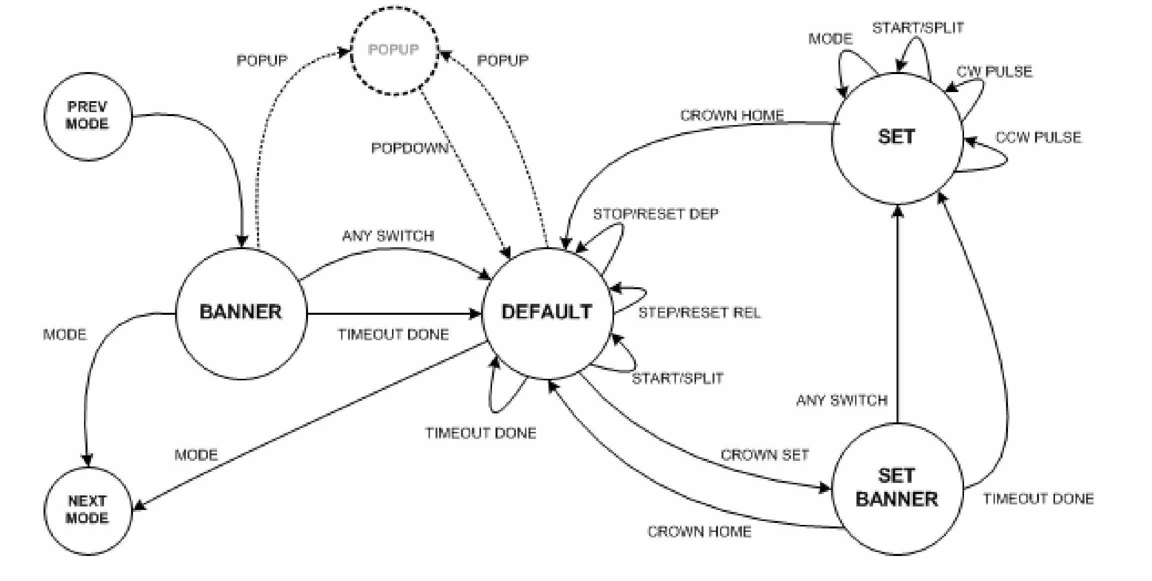


The Set State Handler. This will handle all aspects of setting for the application. The dotted rectangle shows blinking. Each display line represents the fields for setting. This first line shows setting of the counter initial value. The second line shows setting of the count direction.

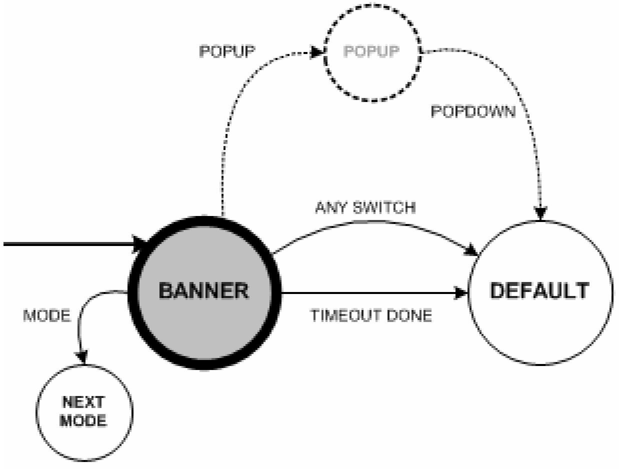
The diagram shown below shows the operations involved to implement the wristapp. Most applications with a setting operation would usually use the basic four states: banner, default, set banner and set state. This allows the wristapp to conform with conventions used in the m851.

The diagram shows the inputs that should be handled by each state handler. This can also serve as a checklist to confirm that all system events are handled.

- Arrows pointing to a state is handled as a State Entry event in the pointed (destination) state.
- Arrows pointing away from a state indicates the event that is processed by the state handler. If it points back to the same state, it means that no state change is required. Lines and arrows pointing to another state indicates that the event should also request for a state change.
- Dotted lines indicate a watch activity that is not controlled by the application such as a popup operation. When a popup is complete through a popdown, the dotted line away from the popup state indicates where it should go back. States with no dotted lines indicates that popups are not allowed to occur. By convention, popups are suspended when the foreground state is either the set banner state or set state.
- Lines going to the state NEXT MODE is handled through a mode change.



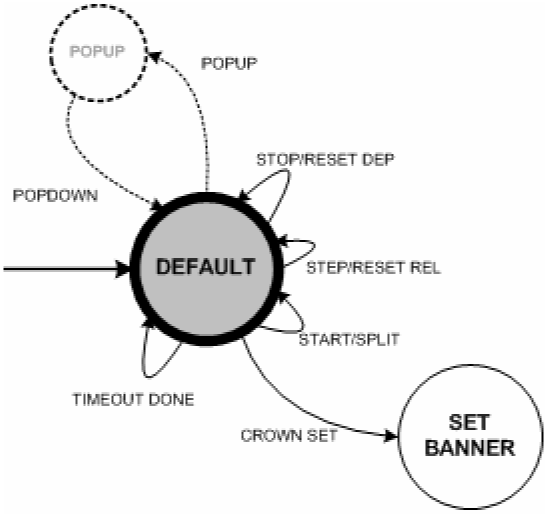
2.2.2 Banner State



The banner state should handle the following cases:

- Handle the system event STATEENTRY and do the following:
  - Allow popups to occur. But popdown should directly proceed to the default state.
  - Request for a 1.5 second hi-res timeout
- Handle the MODESWITCHDEPRESS to go to the next mode.
- Handle STARTSPLITDEPRESS to go to the default state.
- Handle STOPRESETDEPRESS to go to the default state.
- When hi-res timeout expires, proceed to the default state.

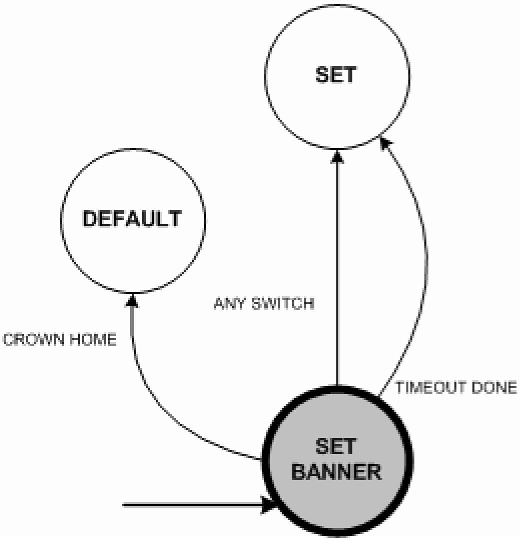
2.2.3 Default State



The default state should handle the following cases:

- Handle the system event STATEENTRY and do the following:
  - Display counter data
- Handle the MODESWITCHDEPRESS to go to the next mode.
- Handle STARTSPLITDEPRESS. This will either increment or decrement the counter. Stop when boundary conditions are reaced.
- Handle STOPRESETDEPRESS to go into a reset operation:
  - Display HOLD TO RESET
  - Allow switch releases to be passed as events
  - Request 2 second hi-res timeout
- Handle STOPRESETRELEASE:
  - Clear display
  - Display counter data
- Handle the event TIMEOUTDONE_HIGHRES :
  - Clear current counter to 0.
  - Display counter data.
- Handle CROWN_SET and request a state change to the set banner state index.

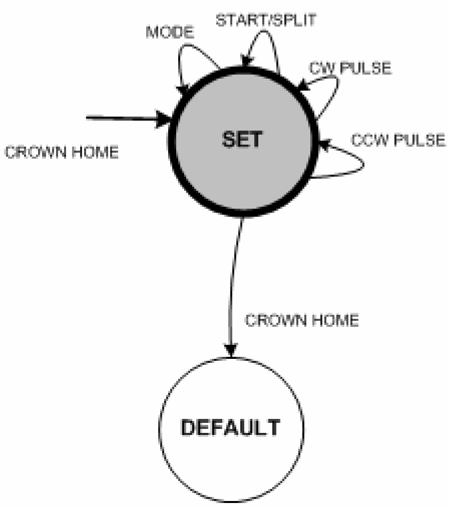
2.2.4 Set Banner State



The set banner state should handle the following cases:

- Handle the system event STATEENTRY and do the following:
  - Do not allow popups to occur.
  - Request for a 1.5 second hi-res timeout
- Handle the MODESWITCHDEPRESS to go to the set state.
- Handle STARTSPLITDEPRESS to go to the set state.
- Handle STOPRESETDEPRESS to go to the set state.
- When hi-res timeout expires, proceed to the set state.
- Handle CROWN_HOME and request a state change to the default state index.

2.2.5 Set State



The set state should handle the following cases:

- Handle the system event STATEENTRY and do the following:
  - Initialize the first setting field position
  - Display current data to be set
  - Setup and request for 4hz blinking
  - Set the system into pulse mode to generate the PULSE events.
- Handle the MODESWITCHDEPRESS to go to the next field setting with wraparound.
- Handle the STOPRESETDEPRESS to go to the next field setting with wraparound.
- Handle the CW_PULSES. Increment counter data (using acceleration) or toggle count direction.
- Handle the CCW_PULSES. Decrement counter data (using acceleration) or toggle count direction.
- Handle CROWN_HOME and request a state change to the default state index (after some data validation).

2.3 State Index

The table below shows the index assigned to each state handler.

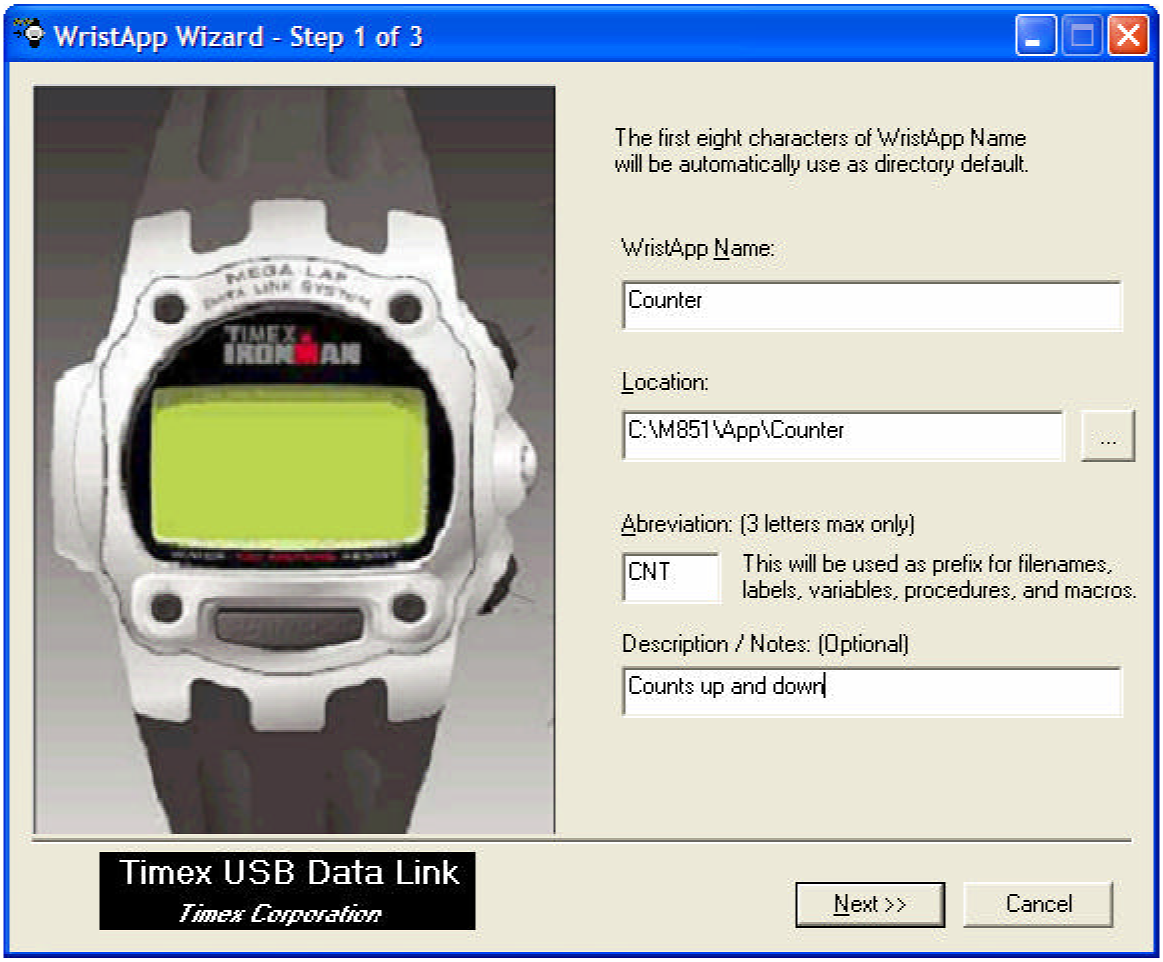
Index	State
0	Banner state index
1	Default state index
2	Set banner state index
3	Set state index

2.4 Using the WristApp Wizard to Create Templates

The WristApp Wizard will facilitate in the creation of the required files for a project. The files generated are complete and can be assembled and linked and downloaded into the watch. The files will serve as a template to be modified to implement the WristApp.

2.4.1 Step 1 of 3

Section	Description
WristApp Name	*Specify the name of the wristapp folder. This is limited to 8 characters due to the limitations imposed on the assembler and linker utilities.*
Location	*Specify the path of the application. By default, the application is stored under the directory C:\m851\app.*
Abbreviation	*The abbreviation code is used to uniquely name the filename, variables, macros, procedures and labels used in the wristapp.*
Description	*Specify the WristApp function.*



The first eight characters of WristApp Name will be automatically use as directory default.

WristApp Name: Counter

Location: C:\M851\App\Counter

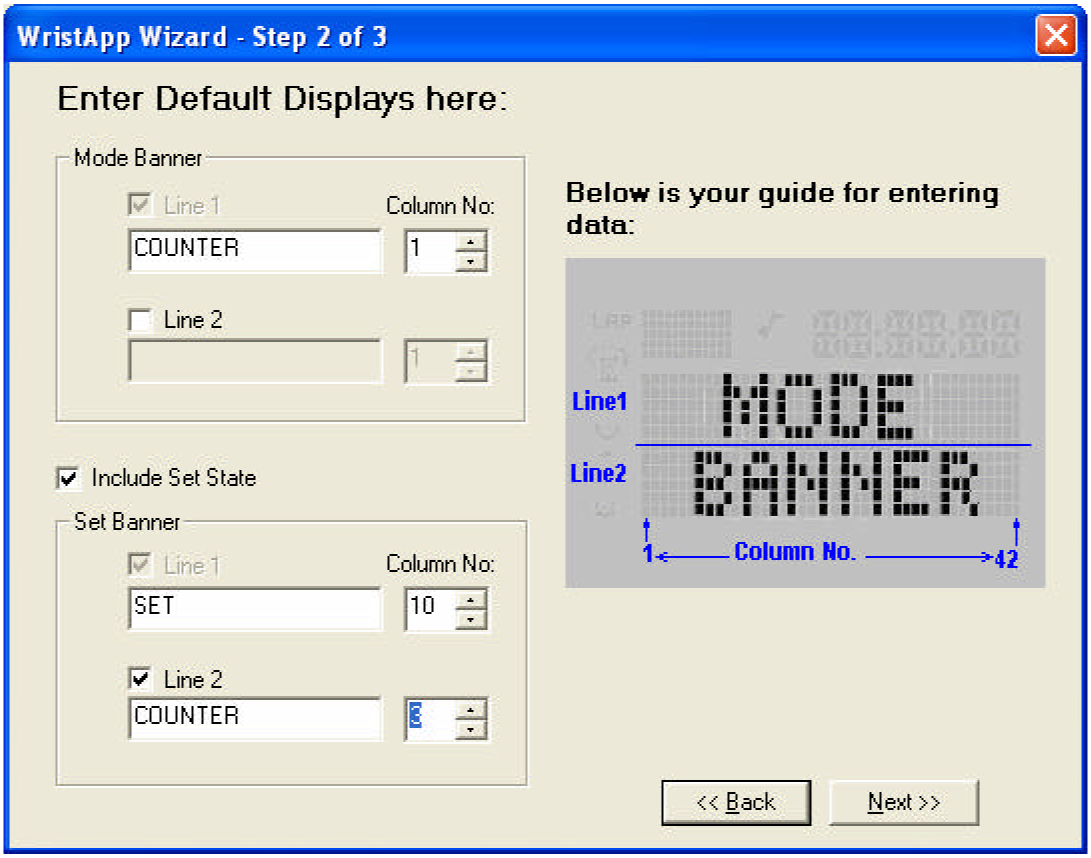
Abreviation: (3 letters max only) CNT

This will be used as prefix for filenames, labels, variables, procedures, and macros.

Description / Notes: (Optional) Counts up and down

2.4.2 Step 2 of 3

Section	Description
Mode Banner	*Specify the text that will be displayed when the WrsitApp becomes the foreground application. The column number is used to center the banner name without resorting to space characters for padding.*
Include Set State	*Check this box if a set state is used in the WristApp. By convention, if a set state is used, there must be a set banner state that describes the setting function.*
Set Banner	*Specify the test that will be displayed when the WristApp enters a setting operation. The column number is used to center the banner name without resorting to space characters for padding.*



WristApp Wizard - Step 2 of 3

Enter Default Displays here:

Mode Banner

☑ Line 1 Column No: 1

COUNTER

☐ Line 2 Column No: 1

☑ Include Set State

Set Banner

☑ Line 1 Column No: 10

SET

☑ Line 2 Column No: 3

COUNTER

Below is your guide for entering data:

2.4.3 Step 3 of 3

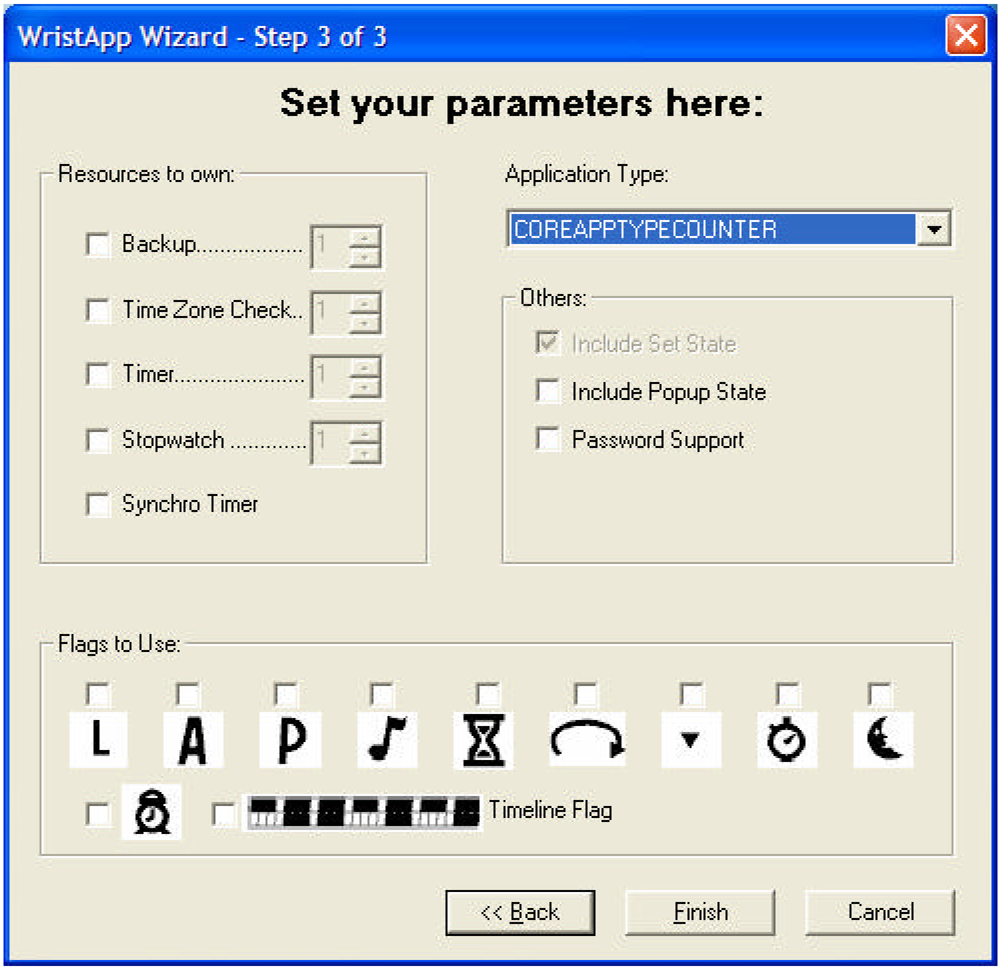
Section	Description
Resource to Own	*Specify the resource type and number required for the WristApp. If a resource is checked, the wizard will automatically create the variable placeholder for the resource index during allocation. This can be found in the XXXvars.h* *Note: The counter wristapp does not use any of the resources provided by the system.*
Application Type	*Select the application type of the WristApp. By default, you can use the COREAPPTYPEGENERIC.* *Certain application types will provide the WristApp will additional benefit specific to an application type. For example, the appointment type wristapp will be passed an event during hour and day rollovers. Alarm type application are called during hour rollovers. Another example is when the primary time zone is modified by the user, the following application types are informed of the change: appointment, occasion and alarm.*
Primary Mode Icon Resource	*Check the primary mode icons used by the WristApp. These icons are used as status information of a wristapp when it is currently active in the*

background or to display WristApp specific information..

For example, the Stopwatch icon is can be used by a chrono wristapp to indicate the state of the chrono: ON if chrono is running, OFF is chrono is stopped.

Another example could be that an alarm wristapp can use the Alarm Clock icon to indicate that an alarm is active and will popup within 12 hours. It could also be blinked to indicate that a backup alarm is pending.

Note: The counter application does not use any primary mode icons.



WristApp Wizard - Step 3 of 3

Set your parameters here:

Resources to own:

☐ Backup................. 1
☐ Time Zone Check.. 1
☐ Timer..................... 1
☐ Stopwatch 1
☐ Synchro Timer

Application Type: COREAPPTYPECOUNTER

Others:

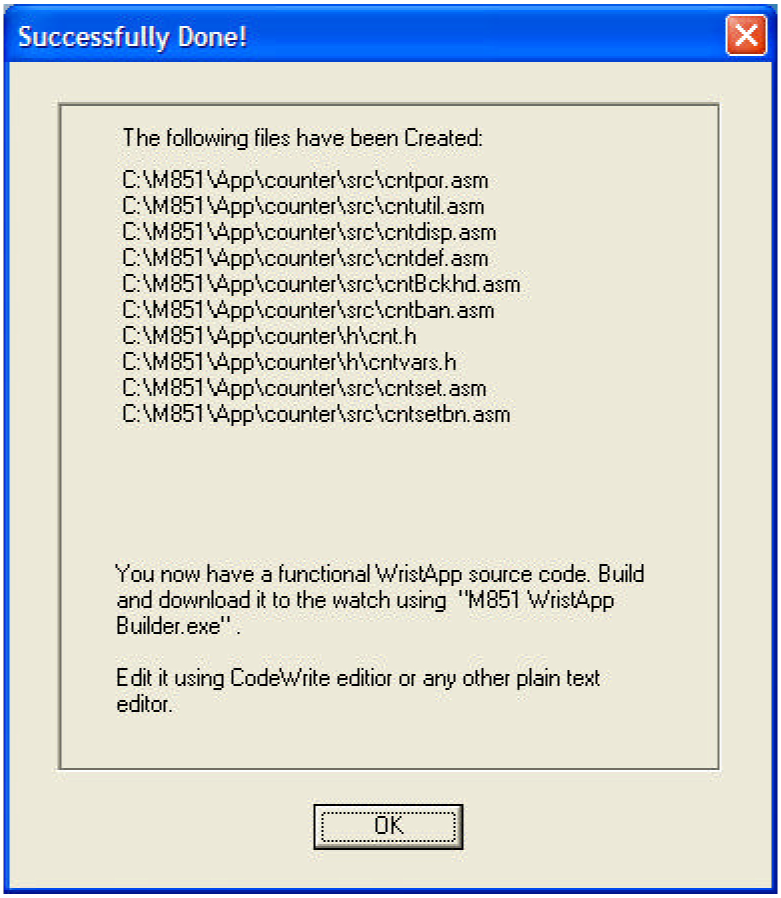
☑ Include Set State
☐ Include Popup State
☐ Password Support

Flags to Use:

☐ L ☐ A ☐ P ☐ ♩ ☐ ⧗ ☐ ↷ ☐ ▾ ☐ ⏱ ☐ ☾
☐ ⏰ ☐ Timeline Flag

2.4.4 File Template Generation

The following screen shows the files being generated by the WristApp wizard ready for modification. This wizrd will also generate the APPNAME.SCR that can be opened by the M851 WristApp Builder utility.



The following files have been Created:

```
C:\M851\App\counter\src\cntpor.asm
C:\M851\App\counter\src\cntutil.asm
C:\M851\App\counter\src\cntdisp.asm
C:\M851\App\counter\src\cntdef.asm
C:\M851\App\counter\src\cntBckhd.asm
C:\M851\App\counter\src\cntban.asm
C:\M851\App\counter\h\cnt.h
C:\M851\App\counter\h\cntvars.h
C:\M851\App\counter\src\cntset.asm
C:\M851\App\counter\src\cntsetbn.asm
```

You now have a functional WristApp source code. Build and download it to the watch using "M851 WristApp Builder.exe".

Edit it using CodeWrite editior or any other plain text editor.

2.5 State Files

The state handlers are to be coded in different files. This will allow the build scripts to properly place the correct state handler code to be loaded from eeprom during execution. In this example, we have the following files:

File	Description
`cntban.asm`	Banner state source file.
`cntdef.asm`	Default state source file.
`cntsetbn.asm`	Set banner state source file.
`cntset.asm`	Set state source file.

2.6 Background Handler

All WristApps are required to handle these system events. In the counter application, most of these required system events are coded with just RET (RETurn from Subroutine) instructions. The background handler will be located at the start of the common section.

Event	Description
`COREEVENT_INIT`	Sent by the system to initialize the application data after a communication session.
`COREEVENT_TASKEXIT`	Sent by the system whenever a voluntary or involuntary (i.e. popup) mode change. This allows

the application to clean up prior to making another application the new foreground application.

`COREEVENT_PEEK` Sent by the system whenever another application requests the mode to display its current data set. For example, the TOD requires an an appointment and occasion type application to support this request.

`COREEVENT_APP_SHUTDOWN_FOR_COMM` Sent by the system when the watch begins communications with the PC. This allows the application to clean up its data and perhaps update the database header ready for upload.

The background handler code will be coded into a separate file. This will allow the build scripts to locate the background handler code to be loaded into the overlay area.

File	**Description**
`cntBckHd.asm`	Background handler source file.

2.7 Parameter File

The M851 requires information about the wristapp so it can be incorporated into the system. Below is the parameter file for the counter application:

```
;==============================================================
; ACB offset mask.
;==============================================================

; Application System Data is located in heap.
; Other ACB entries are located either in ROM or EEPROM.
db      bCOREAppSystemDataOffset


;==============================================================
; Number of resources required.
;==============================================================

db      00h                          ; TOD
db      00h                          ; Backup
db      00h                          ; Time Zone Check
db      00h                          ; Timer Resource
db      00h                          ; Stopwatch Resource
db      00h                          ; Synchro Timer Resource

;==============================================================
; Flag ownership.
;==============================================================

db      0                            ; LCD Flags 1
db      0                            ; LCD Flags 2

;==============================================================
; Heap size requirements.
;==============================================================

dw      0000H                        ; Code
dw      CNTSYSTEMDATASIZE            ; ASD
dw      CNTDATABASEDATASIZE          ; ADD

;==============================================================
; Application Configuration Data Byte.
;==============================================================
db      COREACDEEPROMAPP            ; Code is external.

;============================================================
; Application Unique ID.
;============================================================

db      COREAPPTYPECOUNTER          ; Application type
db      00h                         ; Application instance number

;============================================================
; ACB Parameters.
;============================================================

dw      0000h                       ; ASD address offset
dw      0000h                       ; ADD address offset (no database)
dw      CODESTATEADDRESS            ; App state manager address
dw      CODECOMMONADDRESS           ; App background handler address
dw      lcdBannerMsg_COUNTER        ; App mode name function address
```

Notes:

Code heap size requirement is 0000h.
The utility that will build the wristapp will compute this number automatically. If not using the scripts, this must be the allocation size of the wristapp code in eeprom.

Database heap size requirement.
This value specifies the size of the database being downloaded with the wristapp. The PIM automatically updates this section prior to sending the parameter file to the watch.

The counter wristapp does not have any database stored in external memory. So this value is set at 0x0000.

ASD address offset is 0000h.
All WristApps have offsets of 0000H for its ASD.

ADD address offset is 0000h.
All WristApps uses the EEPROM for database storage. This will always be 0x0000.

The counter does not have database nor store any data in the ADD section whether in internal or external memory.

Use of a label located in ROM: **`lcdBannerMsg_COUNTER`**
The banner message "COUNTER" is already predefined in the M851 OS. This shows that a WristApp is able to execute functions and reference labels embedded in the firmware.

This could well have been a label located in either the common code or in the banner state handler.

The parameter code will be coded into a separate file. This will allow the build scripts to locate the background handler to be used during application download to the watch.

File	Description
`cntpor.asm`	Parameter source file.

2.8 Miscellaneous Files

There are application specific routines that may be used by two or more state handlers. Examples of which are the display routines. Though it can be coded inside the state handler code that uses it, it would be appropriate that it be located in the common section in the WristApp overlay area.

The Counter WristApp requires the following files to be stored in the common section.

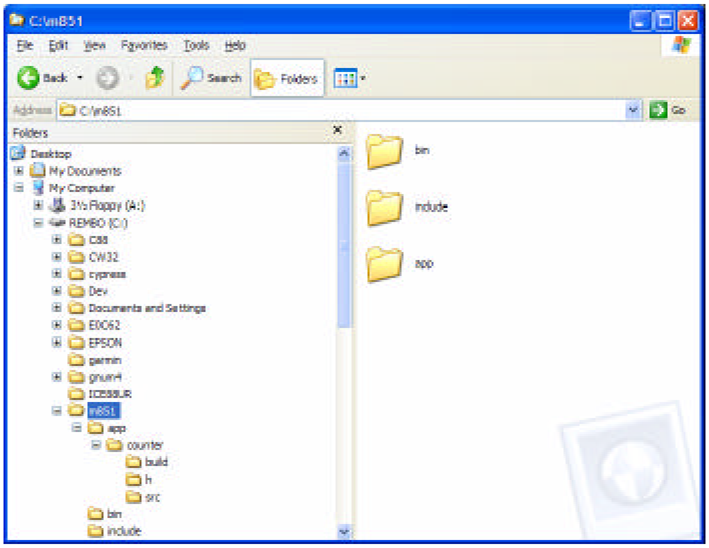
File	Description
`cntdisp.asm`	Display routines for the counter wristapp.
`cntutil.asm`	Utility routines for the counter wristapp.

2.9 Directory Structure

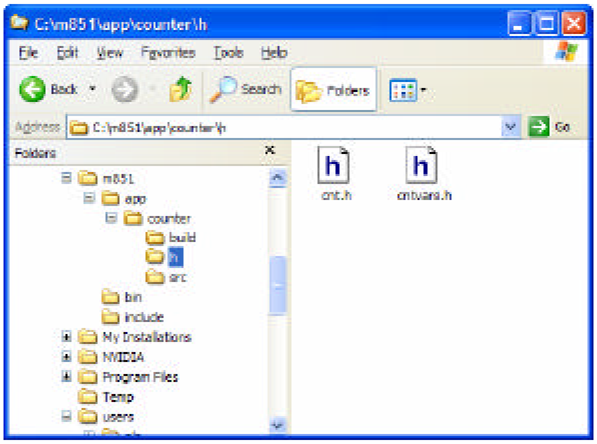
The build scripts requires a specific directory structure to facilitate location of required files. Create the required directories for the application prior to using the build utilities.

- All source files are to be stored under the `C:\M851\APP\`*`appname`*`\SRC` directory.
- All header files are to be stored under the `C:\M851\APP\`*`appname`*`\H` directory.
- All build scripts will be created under the `C:\M851\APP\`*`appname`*`\BUILD` directory.
- Output files during wristapp creation will be in the `C:\M851\APP\`*`appname`*`\BUILD` directory.
- All executable files will be located in the `C:\M851\BIN` directory.
- All the M851 header and macro files will be in the `C:\M851\INCLUDE` directory.
- The assembler, linker and locator executable will be located in the `C:\C88` directory.

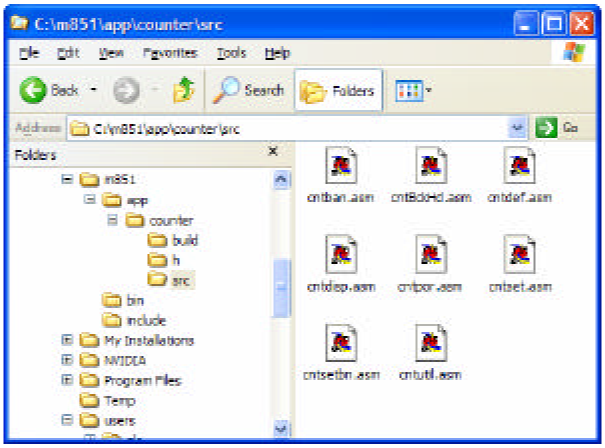
The figure below shows a snapshot of the counter wristapp directory structure:



The figure below shows the file list for the counter wristapp header files:



The figure below shows the file list for the counter wristapp source files:



2.10 *Coding the WristApp*

2.10.1 Header File

Most of the items in the header files are redefinitions to the system equates provided by the M851 OS. The equates are redefined to make the label more descriptive of the operation or function. For example, the switch event equate COREEVENT_SWITCH1DEPRESS in the counter set state handler could be redefined as CNT_CHANGE_TO_NEXT_FIELD_SETTING to indicate a function to change to the next setting field position.

```
;===============================================================================
; STATE REDEFINITIONS
;===============================================================================

CNTBANNERSTATE              equ     COREBANNERSTATE
CNTDEFAULTSTATE             equ     COREDEFAULTSTATE
CNTSETBANNERSTATE           equ     CORESETBANNERSTATE
CNTSETSTATE                 equ     CORESETSTATE

;===============================================================================
; EVENT REDEFINITIONS
;===============================================================================

CNT_STATEENTRY              equ     COREEVENT_STATEENTRY
CNT_TIMEOUTDONELOWRES       equ     COREEVENT_TIMEOUTDONE_LOWRES
```

```
CNT_TIMEOUTDONEHIGHRES      equ     COREEVENT_TIMEOUTDONE_HIGHRES
CNT_TIMEOUTDONESTICKY       equ     COREEVENT_STICKY_TIMEOUTDONE
CNT_ELDEPRESS               equ     COREEVENT_CROWN_EL_DEPRESS
CNT_ELRELEASE               equ     COREEVENT_CROWN_EL_RELEASE
CNT_CROWNHOME               equ     COREEVENT_CROWN_HOME
CNT_CROWNSET                equ     COREEVENT_CROWN_SET1
CNT_CWPULSES                equ     COREEVENT_CW_PULSES
CNT_CCWPULSES               equ     COREEVENT_CCW_PULSES
CNT_CWEDGE                  equ     COREEVENT_CW_EDGE_TRAILING
CNT_CCWEDGE                 equ     COREEVENT_CCW_EDGE_TRAILING
CNT_MODEDEPRESS             equ     COREEVENT_SWITCH1DEPRESS
CNT_STOPRESETDEPRESS        equ     COREEVENT_SWITCH2DEPRESS
CNT_STARTSPLITDEPRESS       equ     COREEVENT_SWITCH3DEPRESS
CNT_MODERELEASE             equ     COREEVENT_SWITCH1RELEASE
CNT_STOPRESETRELEASE        equ     COREEVENT_SWITCH2RELEASE
CNT_STARTSPLITRELEASE       equ     COREEVENT_SWITCH3RELEASE
CNT_POPUPCANCEL             equ     COREEVENT_MELODYPOPUPCANCEL
CNT_DISPLAYUPDATETODRES     equ     COREEVENT_DISPLAY_UPDATE_TODRES
CNT_ICONREFRESH             equ     COREEVENT_ICON_REFRESH
CNT_ANYSWITCHDEPRESS        equ     COREEVENT_ANYSWITCHDEPRESS
CNT_ANYSWITCHRELEASE        equ     COREEVENT_ANYSWITCHRELEASE

;==============================================================================
; SWITCH MASK REDEFINITIONS
;==============================================================================

CNTSWITCHMASK_MODE          equ     bCORESwitch1
CNTSWITCHMASK_STOPRESET     equ     bCORESwitch2
CNTSWITCHMASK_STARTSPLIT    equ     bCORESwitch3
CNTSWITCHMASK_CW            equ     bCORECWSwitch
CNTSWITCHMASK_CCW           equ     bCORECCWSwitch
CNTSWITCHMASK_EL            equ     bCOREELSwitch

CNTSWITCHMASK_CW_CCW_EL     equ     (CNTSWITCHMASK_CW|CNTSWITCHMASK_CCW| _
                                    CNTSWITCHMASK_EL)

;==============================================================================
; HIGH RESOLUTION TIMEOUT DEFINITIONS (Based on 8Hz)
;==============================================================================

CNTHIRESTO_1P5SECONDS       equ     TIMEOUTHIRES_1P5SEC
CNTHIRESTO_2SECONDS         equ     TIMEOUTHIRES_2SEC

;==============================================================================
; MISCELLANEOUS DEFINITIONS
;==============================================================================

; minimum value for the counter data in BCD format
CNTMINDATA                  equ     0000h

; maximum value for the counter data in BCD format
CNTMAXDATA                  equ     0999h
```

2.10.2 Variable File

There is no requirement to separate the contents of the header and variable files. It is coded into separate files for maintenance purposes only.

```
;==============================================================================
; COUNTER APPLICATION SYSTEM DATA
;==============================================================================

; indicates the starting offset for the ASD.  This is always 0x00.
CNTSYSTEMDATASTARTOFFSET        equ     0

CNTFLAGSOFFSET                  equ     0
    bCNTCountDown               equ     00000001B           ; b0 : 0 - Count up
                                                            ;    : 1 - Count down

; Storage for counter data in BCD format.
```

```
CNTDATALOOFFSET                 equ     1
CNTDATAHIOFFSET                 equ     2

; indicates the number of bytes to be allocated in the ASD
CNTSYSTEMDATASIZE               equ     3


;==============================================================================
; COUNTER APPLICATION DATABASE DATA
;==============================================================================

CNTDATABASESTARTOFFSET          equ     CNTSYSTEMDATASIZE
CNTDATABASEDATASIZE             equ     0

;==============================================================================
; FOREGROUND VARIABLE REDEFINITIONS
;==============================================================================

CNTTempFlags                    equ     (COREForegroundCommonBuffer + 0)
    bCNTSetDirection            equ     00000001B  ; b0 : 0 - Change counter data.
                                                   ;    : 1 - Change direction.
```

NOTES:

APPLICATION SYSTEM DATA

Variables stored in this section will maintain its data throughout the life of the application. Access to these variables must be through the index access instructions since the absolute address of the variables is determined only during run-time.

For example:

```
; Set IYReg the address of the counter ASD.
ld      IY, [CORECurrentASDAddress]

; load counter flag value into A register
ld      A, [IY + CNTFLAGSOFFSET]
```

FOREGROUND VARIABLE

Variables stored in this section will be available only if the application is the foreground application. Upon return from a mode change or from a popup, the data stored in this section previously must be assumed to be destroyed.

Compared to data stored in the Application System Data area, variables can be accessed directly. The absolute address of the variable can be determined at design time.

For example:

```
; Load the data that contains the current
;  setting item.
ld      A, [CNTTempFlags]
bit     A, #bCNTSetDirection
jr      Z, cntSetDispAndReqBlinkSetData
```

2.10.3 Banner State Handler

The core provides a common code for the banner state handler. This handles all the requirements for a basic banner state handler.

```
                IF @DEF('SUBROUTINE')
                    UNDEF SUBROUTINE
                ENDIF

                DEFINE  SUBROUTINE      "'cntwaBannerStateManager'"

                GLOBAL  cntwaBannerStateManager

        cntwaBannerStateManager:
                        car     coreCommonBannerStateHandler
                        ret
```

NOTES:

GLOBAL FunctionName	*This will indicate to the assembler and linker system that this function is available to all files compiled in a project.*

IF @DEF('SUBROUTINE')
UNDEF SUBROUTINE
ENDIF
DEFINE SUBROUTINE "'F*unctioNname*'"
	This is a required code prior to a function. The APIs are designed for the M851 OS and would require the SUBROUTINE token to be defined.

2.10.4 Default State Handler

The following is the code for the Counter default state handler.

```
                IF @DEF('SUBROUTINE')
                    UNDEF SUBROUTINE
                ENDIF
                DEFINE  SUBROUTINE      "'cntDefaultStateManager'"

                GLOBAL  cntDefaultStateManager

        cntDefaultStateManager:

                        ; Set IYReg the address of the counter ASD.
                        ld      IY, [CORECurrentASDAddress]

                        ; load in the system event to be processed
                        ld      A, [CORECurrentEvent]

                        ; Check if state entry event.
                        cp      A, #CNT_STATEENTRY
                        jr      Z, cntDefaultStateStateEntryEvent

                        ; Check if start/split depress event.
                        cp      A, #CNT_STARTSPLITDEPRESS
                        jr      Z, cntDefaultStateStartSplitDepressEvent

                        ; Check if stop/reset depress event.
                        cp      A, #CNT_STOPRESETDEPRESS
                        jr      Z, cntDefaultStateStopResetDepressEvent

                        ; Check if stop/reset release event.
                        cp      A, #CNT_STOPRESETRELEASE
                        jr      Z, cntDefaultStateStopResetReleaseEvent

                        ; Check if mode depress event.
                        cp      A, #CNT_MODEDEPRESS
                        jr      Z, cntDefaultStateModeDepressEvent

                        ; Check if timeout hi-res done event.
                        cp      A, #CNT_TIMEOUTDONEHIGHRES
                        jr      Z, cntDefaultStateTimeoutHiResDoneEvent

                        ; Check if crown set event.
                        cp      A, #CNT_CROWNSET
                        jr      NZ, cntDefaultStateExit

                        ;**************************************************************
                        ; CROWN SET
                        ;**************************************************************

                        ; request a state change to the set banner state
                        ld      B, #CNTSETBANNERSTATE
                        CORE_REQ_STATE_CHANGE

        cntDefaultStateExit:
                        ret

        cntDefaultStateStateEntryEvent:

                        ;**************************************************************
                        ; STATE ENTRY
                        ;**************************************************************

                        ; Suspend ring event.  Not used in this state.
                        CORE_SUSPEND_RING_EVENTS

                        ; allow switch releases to be passed to this current state
                        CORE_ENABLE_SWITCH_RELEASE

                        ;-------------------------------------------------------------
                        ; W A R N I N G !!! This is a fall through.  Do not rearrange.
                        ;-------------------------------------------------------------

        cntDefaultSubStateEntry:
        cntDefaultStateStopResetReleaseEvent:

                        ;**************************************************************
                        ; STOP/RESET RELEASE
                        ;**************************************************************

                        push    IY
                        ; display  message count
                        LCD_CLR_DISPLAY
                        LCD_DISP_SEG_MSG_COUNT
                        pop     IY

                        ;--------------------------------------------------------------
                        ; Displays an arrow on the small dot matrix.  The position will
                        ; depend on the count direction.
                        ;--------------------------------------------------------------
                        car     cntDisplayArrowOnSDM

                        ;--------------------------------------------------------------
                        ; Displays the counter data on the main dot matrix using
                        ; large fonts.
                        ;--------------------------------------------------------------
                        jr      cntDisplayCounterData

        cntDefaultStateStartSplitDepressEvent:

                        ;**************************************************************
                        ; START/SPLIT DEPRESS
                        ;**************************************************************

                        ; Cancel current switch release.  Not needed in this state.
                        HW_KBD_CANCEL_CURRENT_SWITCH_RELEASE

                        ; Get the current counter value.
                        ld      HL, IY
                        add     HL, #CNTDATALOOFFSET
                        ld      HL, [HL]
                        ; Load AReg with the counter status flag data and check the
                        ;   counting direction.
                        ld      A, [IY + CNTFLAGSOFFSET]
                        bit     A, #bCNTCountDown
                        jr      Z, cntDefaultStartSplitDepressCountUp

                        ;===============================================================
                        ; COUNT DOWN OPERATION
                        ;===============================================================

                        ; Check whether it is in the minimum value.
                        cp      HL, #CNTMINDATA
                        jr      C, cntDefaultStartSplitDepressExit
                        jr      Z, cntDefaultStartSplitDepressExit

                        ;---------------------------------------------------------------
                        ; Subtract 1 to the counter data.
                        ;---------------------------------------------------------------
                        car     cntSubDataBy1

                        jr      cntDefaultSSDispDataAndReqAlert

        cntDefaultStartSplitDepressCountUp:

                        ;===============================================================
                        ; COUNT UP
                        ;===============================================================

                        ; Check whether it is in the minimum value.
                        cp      HL, #CNTMAXDATA
                        jr      NC, cntDefaultStartSplitDepressExit

                        ;---------------------------------------------------------------
                        ; Add 1 to the counter data.
                        ;---------------------------------------------------------------
                        car     cntAddDataBy1

        cntDefaultSSDispDataAndReqAlert:

                        ;---------------------------------------------------------------
                        ; Displays the counter data on the main dot matrix using
                        ;   large fonts.
                        ;---------------------------------------------------------------
                        car     cntDisplayCounterData

                        ;---------------------------------------------------------------
                        ; Generate alert to indicated that it has successfully
                        ;   decremented/incremented the counter.
                        ;---------------------------------------------------------------
                        AUDSTART_SYSTEM_MELODY AUDSWBEEPMELODY, AUDNOMELODYDONEEVENT

        cntDefaultStartSplitDepressExit:
                        ret

        cntDefaultStateStopResetDepressEvent:

                        ;***************************************************************
                        ; STOP/RESET DEPRESS
                        ;***************************************************************

                        ; Get the current counter value and check whether it is in the
                        ;   minimum value.

                        add     IY, #CNTDATALOOFFSET
                        ld      BA, [IY]

                        cp      BA, #CNTMINDATA
                        jr      Z, cntDefaultStopResetDepressExit

                        ; Not yet in its minimum.
                        ; Request 2sec timeout.
                        CORE_REQ_TIMEOUT_HIRES  CNTHIRESTO_2SECONDS

                        LCD_CLR_DISPLAY
                        LCD_DISP_SMALL_DM_MSG_HOLD_TO_RESET

        cntDefaultStopResetDepressExit:
                        ret

        cntDefaultStateModeDepressEvent:

                        ;*************************************************************
                        ; MODE DEPRESS
                        ;*************************************************************
                        CORE_REQ_MODE_CHANGE_NEXT
                        ret

        cntDefaultStateTimeoutHiResDoneEvent:

                        ;*************************************************************
                        ; TIMEOUT DONE HI-RES
                        ;*************************************************************

                        ; Cancel current switch release.  Not needed in this state.
                        HW_KBD_CANCEL_CURRENT_SWITCH_RELEASE

                        AUDSTART_SYSTEM_MELODY AUDSWBEEPMELODY, AUDNOMELODYDONEEVENT

                        ; Clear counter data.
                        ld      A, #0
                        ld      [IY + CNTDATALOOFFSET], A
                        ld      [IY + CNTDATAHIOFFSET], A

                        ; Redisplay everything.
                        jr      cntDefaultSubStateEntry
```

2.10.5 Set Banner State Handler

The core provides a common code for the set banner state handler. It requires application specific code to handle what to display during state entry. The rest of the code handles the basic requirements for the set banner state handler.

```
        IF @DEF('SUBROUTINE')
            UNDEF SUBROUTINE
        ENDIF
        DEFINE  SUBROUTINE      "'cntSetBannerStateManager'"

        GLOBAL  cntSetBannerStateManager

cntSetBannerStateManager:

                ; Get the event to be processed.
                ld      A,  [CORECurrentEvent]

                ; Check if State Entry Event.
                cp      A,  #CNT_STATEENTRY
                jr      NZ, utlSetBannerStateManager

                ;*************************************************************
                ; STATE ENTRY
                ;*************************************************************

                LCD_DISP_SMALL_DM_MSG_SET_COUNTER
                jr     utlSetBannerStateManager
```

2.10.6 Set State Handler

The following is the code for the Counter set state handler.

```
        IF @DEF('SUBROUTINE')
            UNDEF SUBROUTINE
        ENDIF

        DEFINE  SUBROUTINE      "'cntSetStateManager'"

        GLOBAL  cntSetStateManager

cntSetStateManager:

                ; Set IYReg the address of the counter ASD.
                ld      IY, [CORECurrentASDAddress]

                ld      A, [CORECurrentEvent]

                cp      A, #CNT_STATEENTRY
                jr      Z, cntSetStateStateEntryEvent

                cp      A, #CNT_MODEDEPRESS
                jr      Z, cntSetStateModeDepressEvent

                cp      A, #CNT_STOPRESETDEPRESS
                jr      Z, cntSetStateStopResetDepressEvent

                cp      A, #CNT_CWPULSES
                jr      Z, cntSetStateCWPulseEvent

                cp      A, #CNT_CCWPULSES
                jr      Z, cntSetStateCCWPulseEvent

                cp      A, #CNT_CROWNHOME
                jr      NZ, cntSetStateExit

                ;**************************************************************
                ; CROWN HOME
                ;**************************************************************

                ld      B, #CNTDEFAULTSTATE
                CORE_REQ_STATE_CHANGE

cntSetStateExit:
                ret

cntSetStateStateEntryEvent:

                ;**************************************************************
                ; STATE ENTRY
                ;**************************************************************

                ; Enable pulse mode to change values.
                CORE_ENABLE_PULSE_MODE

                ; Mask start/split key.  This event is not needed.
                CORE_MASK_KEYS  (CNTSWITCHMASK_STARTSPLIT | CNTSWITCHMASK_EL)

                ; Clear the bit indicating that the first set item is the counter
                ;   data.
                ld      HL, #CNTTempFlags
                and     [HL], #@LOW(~bCNTSetDirection)

                ;--------------------------------------------------------------
                ; Refresh the display and request blinking on the editable field.
                ; Destroys BAReg, HLReg, IXReg.
                ; Input: IYReg - ASD address.
                ;--------------------------------------------------------------
                jr      cntSetRedisplayAndReqBlink  ; **EXTERNAL JUMP
```

```
cntSetStateModeDepressEvent:
cntSetStateStopResetDepressEvent:

                ;**************************************************************
                ; STOP/RESET & MODE DEPRESS
                ;**************************************************************

                ; Load the address to HLReg and toggle the set direction flag.
                ld      HL, #CNTTempFlags
                xor     [HL], #bCNTSetDirection

                ;--------------------------------------------------------------
                ; Clear the entire display.
                ;--------------------------------------------------------------
                LCD_CLR_DISPLAY

                ;--------------------------------------------------------------
                ; Refresh the display and request blinking on the editable field.
                ;--------------------------------------------------------------
                jr      cntSetRedisplayAndReqBlink  ; **EXTERNAL JUMP

cntSetStateCWPulseEvent:

                ;**************************************************************
                ; CW PULSE (Increment Field Value)
                ;**************************************************************

                ; Check the item to be set.
                ld      HL, #CNTTempFlags
                bit     [HL], #bCNTSetDirection
                jr      NZ, cntSetStateToggleDirection

                ;--------------------------------------------------------------
                ; Add counter data by acceleration value.
                ; Destroys BAReg, IXReg, HLReg.
                ; Input: IYReg - ASD address
                ;        COREEventArgument - Number of pulses
                ;--------------------------------------------------------------
                car     cntAddDataByAcceleration

                ;--------------------------------------------------------------
                ; Refresh the display and request blinking on the editable field.
                ; Destroys BAReg, HLReg, IXReg.
                ; Input: IYReg - ASD address.
                ;--------------------------------------------------------------
                jr      cntSetRedisplayAndReqBlink  ; **EXTERNAL JUMP

cntSetStateCCWPulseEvent:

                ;**************************************************************
                ; CCW PULSE (Decrement Field Value)
                ;**************************************************************

                ; Check the item to be set.
                ld      HL, #CNTTempFlags
                bit     [HL], #bCNTSetDirection
                jr      NZ, cntSetStateToggleDirection

                ;--------------------------------------------------------------
                ; Subtract counter data by acceleration value.
                ; Destroys BAReg, IXReg.
                ; Input: IYReg - ASD address
                ;        COREEventArgument - Number of pulses
                ;--------------------------------------------------------------
                car     cntSubDataByAcceleration

                ;--------------------------------------------------------------
                ; Refresh the display and request blinking on the editable field.
                ; Destroys BAReg, HLReg, IXReg.
                ; Input: IYReg - ASD address.
```

```
                        ;---------------------------------------------------------------
                        jr      cntSetRedisplayAndReqBlink  ; **EXTERNAL JUMP

        cntSetStateToggleDirection:

                        ; Toggle count-up/countdown bit.
                        ld      A, [IY + CNTFLAGSOFFSET]
                        xor     A, #bCNTCountDown
                        ld      [IY + CNTFLAGSOFFSET], A

                        ;---------------------------------------------------------------
                        ; Clear line 2 only so that the display would not look like
                        ;   garbage when changing from "DOWN" to "UP".
                        ; Destroys AReg, IXReg.
                        ;---------------------------------------------------------------
                        LCD_CLR_MAIN_DM_LINE2

                        ;---------------------------------------------------------------
                        ; Refresh the display and request blinking on the editable field.
                        ; Destroys BAReg, HLReg, IXReg.
                        ; Input: IYReg - ASD address.
                        ;---------------------------------------------------------------
                        jr      cntSetRedisplayAndReqBlink  ; **EXTERNAL JUMP
```

2.10.7 Background Handler

The following code handles the events passed by the M851 OS to the counter wristapp background handler. Only the INIT event is seen processed here. The `TASKEXIT`, `PEEK`, and `APP_SHUTDOWN_FOR_COMM` are handled only as return instructions.

```
                        IF @DEF('SUBROUTINE')
                            UNDEF SUBROUTINE
                        ENDIF
                        DEFINE  SUBROUTINE      "'cntBackgroundHandler'"

                        GLOBAL  cntBackgroundHandler

        cntBackgroundHandler:

                        ; Load the event to be process to AReg.
                        ld      A, [COREBackgroundEvent]

                        ; Check if INIT event.
                        cp      A, #COREEVENT_INIT
                        jr      NZ, cntBackgroundProcessExit

        cntBackgroundInitEvent:

                        ;***************************************************************
                        ; INITIALIZATION THROUGH COMM MODE
                        ;***************************************************************

                        ;---------------------------------------------------------------
                        ; Counter initial data.
                        ;   Data - 0
                        ;   Count up
                        ;---------------------------------------------------------------

                        ld      A, #0
                        ld      IY, [COREInitializationASDAddress]
                        ld      [IY + CNTFLAGSOFFSET], A
                        ld      [IY + CNTDATALOOFFSET], A
                        ld      [IY + CNTDATAHIOFFSET], A

        cntBackgroundProcessExit:
                        ret
```

2.10.8 Display Routines

The following is the code for the Counter display routines.

cntDisplayArrowOnSDM
cntDisplayArrowDownOnSDM
cntDisplayArrowUpOnSDM

```
        IF @DEF('SUBROUTINE')
            UNDEF SUBROUTINE
        ENDIF

        DEFINE  SUBROUTINE      "'cntDisplayArrowOnSDM'"

        GLOBAL  cntDisplayArrowOnSDM
        GLOBAL  cntDisplayArrowDownOnSDM
        GLOBAL  cntDisplayArrowUpOnSDM

cntDisplayArrowOnSDM:

                ; Get the status flags.
                ld      A, [IY + CNTFLAGSOFFSET]

                ; Check the counting direction.
                bit     A, #bCNTCountDown
                jr      Z, cntDisplayArrowUpOnSDM

cntDisplayArrowDownOnSDM:

                ; Load the character to be displayed.
                ld      L, #DM5_DOWNARROW

                jr      cntDispArrowOnSDMDisplay

cntDisplayArrowUpOnSDM:

                ; Load the character to be displayed.
                ld      L, #DM5_UPARROW

cntDispArrowOnSDMDisplay:

                ;---------------------------------------------------------------
                ; Display proportional width character.
                ; Destroys BAReg, HLReg, IXReg.
                ; Input: LReg - Characer to be displayed.
                ;        IXReg- Starting DM column.
                ;---------------------------------------------------------------
                ld      IX, #LCDUPPERDMCOL1
                LCD_DISP_SMALL_PROP_WIDTH_DM_CHAR
                ret
```

cntDisplayCounterData

```
        IF @DEF('SUBROUTINE')
            UNDEF SUBROUTINE
        ENDIF
        DEFINE  SUBROUTINE      "'cntDisplayCounterData'"

        GLOBAL  cntDisplayCounterData

cntDisplayCounterData:

                ;---------------------------------------------------------------
                ; Display a large-font, 3-digit DM data with zero suppression
                ;   on leading digit positions.
                ; Destroys BAReg, HLReg, IXReg.
                ; Input:  BReg - 100's digit BCD data.
                ;         AReg - Packed 10's and 1's digit BCD data.
                ;         IXReg- Starting DM column.
                ;--------------------------------------------------------------
                ld      IY, [CORECurrentASDAddress]
                ld      A, [IY + CNTDATALOOFFSET]
                ld      B, [IY + CNTDATAHIOFFSET]
                ld      IX, #LCDBIGCHARDMCOL8
                LCD_DISP_BIG_3DIGIT_DM_DATA_NO_LSD_SUP
                ret
```

cntDisplayCountDirection

```
        IF @DEF('SUBROUTINE')
            UNDEF SUBROUTINE
        ENDIF
        DEFINE  SUBROUTINE      "'cntDisplayCountDirection'"

        GLOBAL  cntDisplayCountDirection

cntDisplayCountDirection:

                ; Get the status flags and check the counting direction.
                ld      IY, [CORECurrentASDAddress]
                ld      A, [IY + CNTFLAGSOFFSET]
                bit     A, #bCNTCountDown
                jr      Z, cntDisplayDirectionArrowUp

                ; Display "COUNT DOWN" on the main DM.
                LCD_DISP_SMALL_DM_MSG_COUNT_DOWN

                ; Display arrow down on SDM.
                jr      cntDisplayArrowDownOnSDM    ; **EXTERNAL JUMP

cntDisplayDirectionArrowUp:

                ; Display "COUNT UP" on the main DM.
                LCD_DISP_SMALL_DM_MSG_COUNT_UP

                ; Display arrow up on SDM.
                jr      cntDisplayArrowUpOnSDM      ; **EXTERNAL JUMP
```

cntClearL2AndSDM

```
        IF @DEF('SUBROUTINE')
            UNDEF SUBROUTINE
        ENDIF
        DEFINE  SUBROUTINE      "'cntClearL2AndSDM'"

        GLOBAL  cntClearL2AndSDM

cntClearL2AndSDM:

                ; Clear SDM.
                LCD_CLEAR_UPPER_DM

                ; Clear line 2.
                LCD_CLR_MAIN_DM_LINE2
                ret
```

cntSetRedisplayAndReqBlink

```
        IF @DEF('SUBROUTINE')
            UNDEF SUBROUTINE
        ENDIF
        DEFINE  SUBROUTINE      "'cntSetRedisplayAndReqBlink'"

        GLOBAL  cntSetRedisplayAndReqBlink

cntSetRedisplayAndReqBlink:

                CORE_REQ_BLINK_4HZ
                ; Load the data that contains the current setting item.
                ld      A, [CNTTempFlags]
                bit     A, #bCNTSetDirection
                jr      Z, cntSetDispAndReqBlinkSetData

                ; Change the couting direction.

                ; Display "COUNT DOWN" or "COUNT UP" and "Arrow Down" or
                ;   "Arrow Up" on main DM and SDM respectively.
                car     cntDisplayCountDirection

                ; Setup the routines to be called for blinking.
                LCD_WRITE_4HZ_GEN_BLINK_DISP_ROUTINE_ADDR cntDisplayCountDirection
                LCD_WRITE_4HZ_GEN_BLINK_CLR_ROUTINE_ADDR cntClearL2AndSDM
                ret

cntSetDispAndReqBlinkSetData:

                ; Change the counter value.

                ; Displays the counter data on the main dot matrix using large
                ;   fonts.
                car     cntDisplayCounterData

                ; Display "SET" on 9 segment.
                LCD_DISP_SEG_MSG_SET

                ; Setup the routines to be called for blinking.
                LCD_WRITE_4HZ_GEN_BLINK_DISP_ROUTINE_ADDR   cntDisplayCounterData
                LCD_WRITE_4HZ_GEN_BLINK_CLR_ROUTINE_ADDR    lcdClearMainDM
                ret
```

2.10.9 Utility Routines

The following is the code for the Counter utility routines.

cntAddDataBy1

```
        IF @DEF('SUBROUTINE')
            UNDEF SUBROUTINE
        ENDIF
        DEFINE  SUBROUTINE      "'cntAddDataBy1'"

        GLOBAL  cntAddDataBy1

cntAddDataBy1:

                push    SC

                ; Use decimal addition.
                UTL_DECIMAL_MATH_MODE

                ; Value to be added to the counter data.
                ld      A, #01h

                ; Compute the new counter data.
                ; Popping of SCReg is done inside the routine.
                jr      cntAddDataBy1EntryPoint
```

cntAddDataByAcceleration

```
        IF @DEF('SUBROUTINE')
            UNDEF SUBROUTINE
        ENDIF

        DEFINE  SUBROUTINE      "'cntAddDataByAcceleration'"
        GLOBAL  cntAddDataByAcceleration

cntAddDataByAcceleration:                           ; **SUBROUTINE
cntAddDataByAcceleration

                push    SC

                ; Use decimal addition.
                UTL_DECIMAL_MATH_MODE

                ;--------------------------------------------------------------
                ; Determine the acceleration factor for COREEventArgument and
                ;   write factor into AReg.
                ;--------------------------------------------------------------

                ;--------------------------------------------------------------
                ; Get starting address into the acceleration table then subtract
                ;   it by 1 to get the exact acceleration data.  Take note that
                ;   the least number of pulses that the system will send is 1.
                ;--------------------------------------------------------------
                ld      IX, #utlAccelerationTable1Min - 1

                ; Get the number of pulses.
                ld      L, [COREEventArgument]

                ; Get the acceleration factor.
                ld      A, [IX + L]

cntAddDataBy1EntryPoint:
                ;--------------------------------------------------------------
                ; Note for using this as the entry point.
                ;      AReg - Value to be added to the current counter.
                ;      IYReg- Counter ASD address.
                ;      SCReg should be pushed.
                ;      bDecimalFlag should be set.
                ;--------------------------------------------------------------

                push    IY

                ; Set HLReg and IYReg to point to the data low address.
                add     IY, #CNTDATALOOFFSET
                ld      HL, IY

                ; Increment the counter.
                add     [HL], A
                inc     HL
                adc     [HL], #0

                ; Get the current counter data.
                ld      HL, [IY]

                ;--------------------------------------------------------------
                ; Check if counter data exceeds its maximum.  If it exceeds
                ;   then compute for the excess data so that it would look
                ;   like it has wraparound.
                ;--------------------------------------------------------------
                cp      HL, #CNTMAXDATA
                jr      C, cntAddDataExit
                jr      Z, cntAddDataExit

                ld      HL, IY
                sub     [HL], #@LOW(CNTMAXDATA+1)
                inc     HL
                sbc     [HL], #@HIGH(CNTMAXDATA)

cntAddDataExit:

                pop     IY
                pop     SC
                ret
```

cntSubDataBy1

```
        IF @DEF('SUBROUTINE')
            UNDEF SUBROUTINE
        ENDIF
        DEFINE  SUBROUTINE      "'cntSubDataBy1'"

        GLOBAL  cntSubDataBy1

cntSubDataBy1:                                      ; **SUBROUTINE cntSubDataBy1

                push    SC

                ; Use decimal addition.
                UTL_DECIMAL_MATH_MODE

                ; Value to be subtracted to the counter data.
                ld      A, #01h

                ; Compute the new counter data.
                ; Popping of SCReg is done inside the routine.
                jr      cntSubDataBy1EntryPoint
```

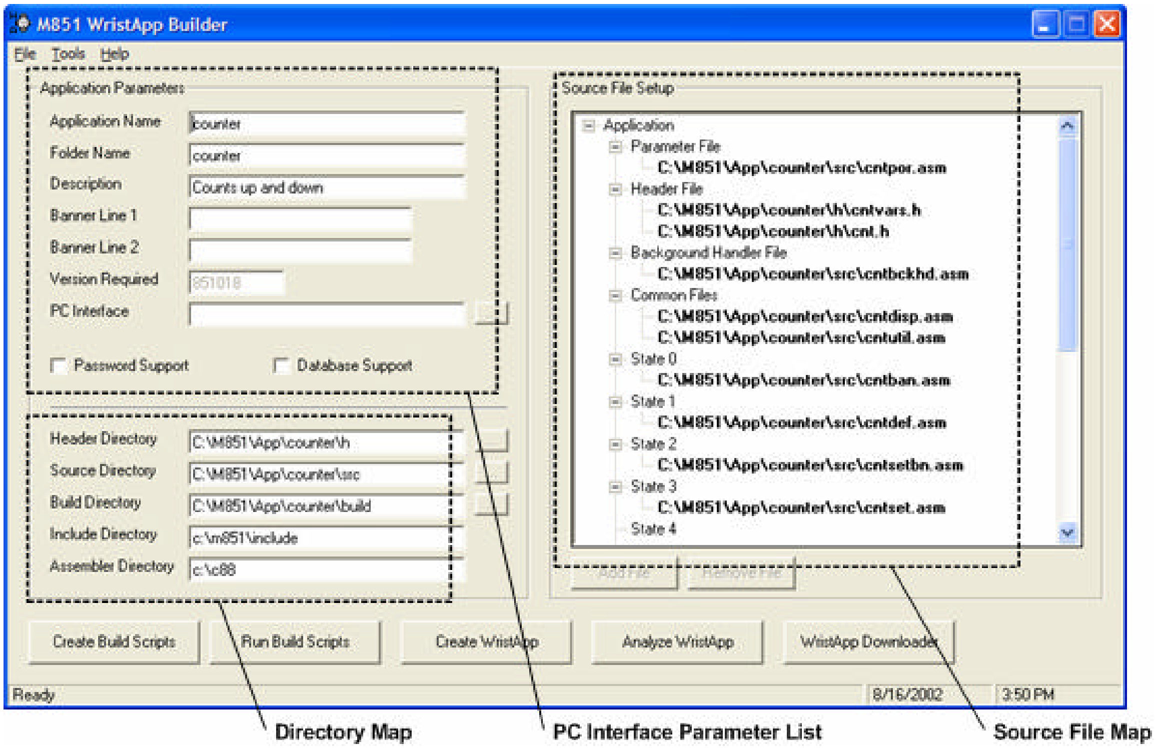
cntSubDataByAcceleration

```
        IF @DEF('SUBROUTINE')
            UNDEF SUBROUTINE
        ENDIF
        DEFINE  SUBROUTINE      "'cntSubDataByAcceleration'"

        GLOBAL  cntSubDataByAcceleration

cntSubDataByAcceleration:

                push    SC

                ; Use decimal addition.
                UTL_DECIMAL_MATH_MODE

                ;---------------------------------------------------------------
                ; Determine the acceleration factor for COREEventArgument and
                ;   write factor into AReg.
                ;---------------------------------------------------------------

                ;---------------------------------------------------------------
                ; Get starting address into the acceleration table then subtract
                ;   it by 1 to get the exact acceleration data.  Take note that
                ;   the least number of pulses that the system will send is 1.
                ;---------------------------------------------------------------
                ld      IX, #utlAccelerationTable1Min - 1

                ; Get the number of pulses.
                ld      L, [COREEventArgument]

                ; Get the acceleration factor.
                ld      A, [IX + L]

cntSubDataBy1EntryPoint:
                ;---------------------------------------------------------------
                ; Note for using this as the entry point.
                ;      AReg - Value to be added to the current counter.
                ;      IYReg- Counter ASD address.
                ;      SCReg should be pushed.
                ;      bDecimalFlag should be set.
                ;---------------------------------------------------------------

                push    IY

                ; Set HLReg and IYReg to point to the data low address.
                add     IY, #CNTDATALOOFFSET
```

```
                        ld      HL, IY

                        ; Decrement the counter.
                        sub     [HL], A
                        inc     HL
                        sbc     [HL], #0

                        ; Get the current counter data.
                        ld      HL, [IY]

                        ;----------------------------------------------------------------
                        ; Check if counter data exceeds its minumum.  If it exceeds
                        ;   then compute for the excess data so that it would look
                        ;   like it has wraparound.
                        ;----------------------------------------------------------------
                        cp      HL, #CNTMAXDATA
                        jr      C, cntSubDataExit
                        jr      Z, cntSubDataExit

                        ld      HL, IY
                        add     [HL], #@LOW(CNTMAXDATA + 1)
                        inc     HL
                        adc     [HL], #@HIGH(CNTMAXDATA)

        cntSubDataExit:

                        pop     IY
                        pop     SC
                        ret
```

2.11 Creating the WristApp

This section will guide you to a series of steps to build a WristApp. At this point, it is assumed that all files required for the wristapp has been coded (and hopefully reviewed). A WristApp Builder utility is provided in the SDK package that will facilitate the process. The utility is located in the `C:\M851\BIN` directory.



Application Name: counter
Folder Name: counter
Description: Counts up and down
Banner Line 1:
Banner Line 2:
Version Required: 851018
PC Interface:
□ Password Support □ Database Support

Header Directory: C:\M851\App\counter\h
Source Directory: C:\M851\App\counter\src
Build Directory: C:\M851\App\counter\build
Include Directory: c:\m851\include
Assembler Directory: c:\c88

```
Application
    Parameter File
        C:\M851\App\counter\src\cntpor.asm
    Header File
        C:\M851\App\counter\h\cntvars.h
        C:\M851\App\counter\h\cnt.h
    Background Handler File
        C:\M851\App\counter\src\cntbckhd.asm
    Common Files
        C:\M851\App\counter\src\cntdisp.asm
        C:\M851\App\counter\src\cntutil.asm
    State 0
        C:\M851\App\counter\src\cntban.asm
    State 1
        C:\M851\App\counter\src\cntdef.asm
    State 2
        C:\M851\App\counter\src\cntsetbn.asm
    State 3
        C:\M851\App\counter\src\cntset.asm
    State 4
```

Ready 8/16/2002 3:50 PM

Directory Map

PC Interface Parameter List

Source File Map

Section	Description
Directory Map	*Shows the locations of source files, executables, include files, and assembler files.*
PC Interface Parameter List	*Data source to fill out the *.APP file used by the PIM to download a WristApp to a watch.*
Source File Map	*A hierarchal view of the files associations to the actual wristapp function.*

NOTE: The information displayed in the utility is stored in a file APPNAME.SCR. The file is created when the build scripts are generated or it was saved through the **File\Save** menu. The file is stored in the build directory of the application.

2.11.1 PC Interface Parameter List

Fill up all the required information in Application Parameter section.

Field	Description
Application Name	*Descriptive name of the application.*
Folder Name	*Indicates the application folder name. Entering data in the Folder Name text box will automatically fill up the required entries in the Directory Map section.*
Description	*A brief description of the application.*
Banner Line 1 **Banner Line 2**	*Mode banner message to be display in line either 1 and/or line 2. A blank entry in these two sections will tell the application to use the mode banner name indicated in the parameter file.*
Version Required	*Indicates the M851 firmware version that the wristapp is referencing.*
PC Interface	*Indicates the PC Interface of the wristapp. This interface will handle any special requirements of an application prior to download to the watch. This utility is also responsible for setting up the database that an application will require.*
Password Support	*Indicates if the wristapp is designed to support password protection that can be checked by the PIM.*
Database Support	*Indicates if the wristapp requires a database to be downloaded with the WristApp. This is checked by the PIM.*

2.11.2 Source File Map

Add the files associated with the different application sections.

Section	Description
Parameter File	*Application Parameter List file.*
Header File	*List of header files specific to the application. The variable file is to be located in this list.*
Background Handler File	*Background Handler source file. The background handler routine is*

located as the first module in the common section.

Common Files *List of files to be located in the common section of the overlay area. Generally, the utility files and the display source files are located in this list.*

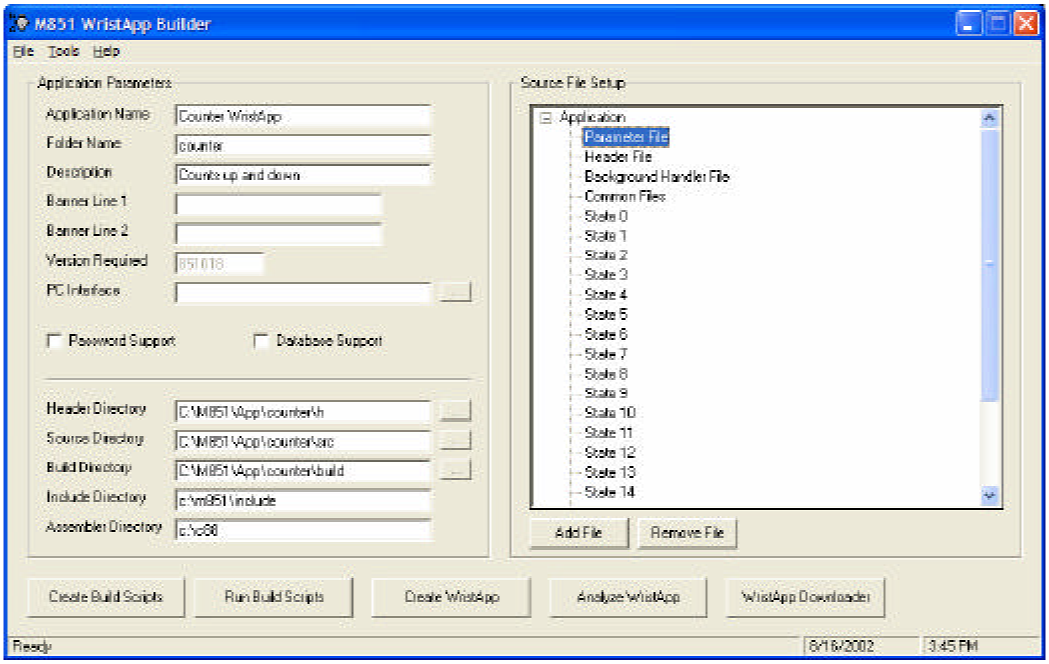
State *n* File *Stores the source file of an associated state index.*

There are two procedures in adding files into each section of the Source File Map.

- Using the Add File button;
- Using Drag & Drop method from File Explorer.

Adding a File using the Add File button.

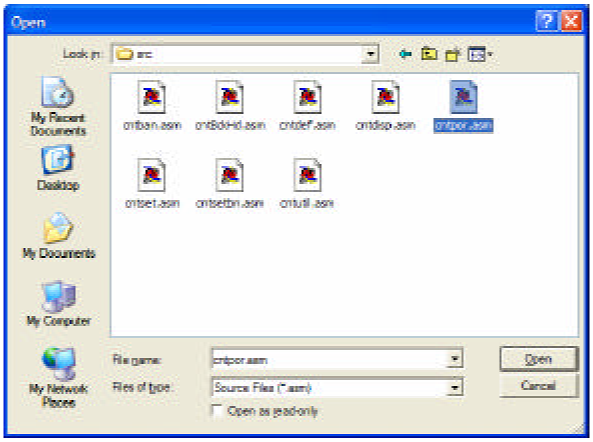
Click on a section where the new file is to be added (the figure shows the "Parameter File" being selected. Then click on the "**Add File**" button to open up the Open dialog window.



Application Name: Counter WristApp
Folder Name: counter
Description: Counts up and down
Banner Line 1:
Banner Line 2:
Version Required: 851018
PC Interface:
□ Password Support □ Database Support
Header Directory: C:\M851\App\counter\h
Source Directory: C:\M851\App\counter\src
Build Directory: C:\M851\App\counter\build
Include Directory: c:\m851\include
Assembler Directory: c:\c68

Ready 8/16/2002 3:45 PM

Select the file to be added in the section and click **Open**. The figure below shows the file "CNTPOR.ASM" selected.



Open

Look in: src

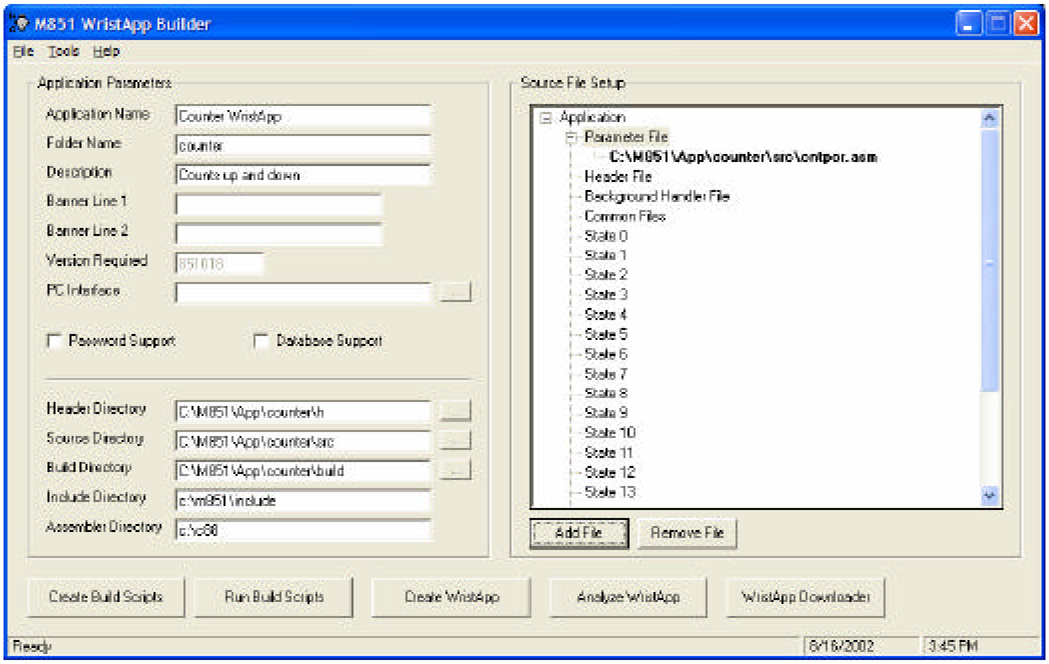
cntban.asm cntBckHd.asm cntdef.asm cntdisp.asm cntpor.asm
cntset.asm cntsetbn.asm cntutil.asm

File name: cntpor.asm

Files of type: Source Files (*.asm)

□ Open as read-only

After this operation, the file CNTPOR.ASM will be added under the “Parameter File” section. See figure below.



M851 WristApp Builder

Application Parameters

Application Name: Counter WristApp

Folder Name: counter

Description: Counts up and down

Banner Line 1:

Banner Line 2:

Version Required: 851018

PC Interface:

□ Password Support □ Database Support

Header Directory: C:\M851\App\counter\h

Source Directory: C:\M851\App\counter\src

Build Directory: C:\M851\App\counter\build

Include Directory: c:\m851\include

Assembler Directory: c:\c88

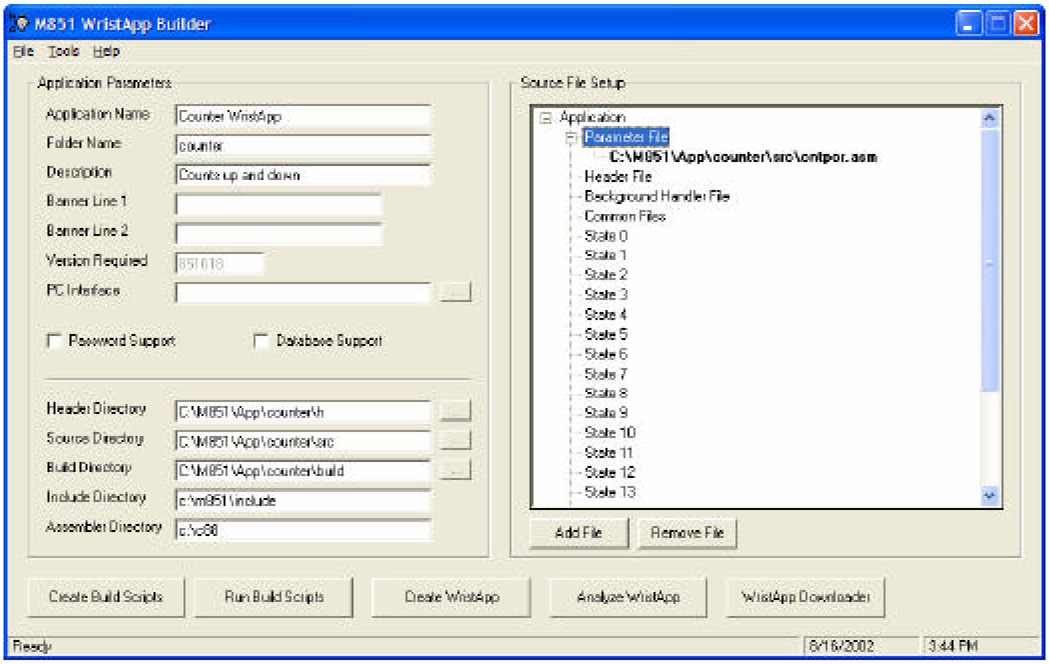
Source File Setup

```
Application
  Parameter File
    C:\M851\App\counter\src\cntpor.asm
  Header File
  Background Handler File
  Common Files
  State 0
  State 1
  State 2
  State 3
  State 4
  State 5
  State 6
  State 7
  State 8
  State 9
  State 10
  State 11
  State 12
  State 13
```

Ready 8/16/2002 3:45 PM

Adding a file using File Explorer.

Click on a section where the new file is to be added (the figure shows the “Header File” being selected.

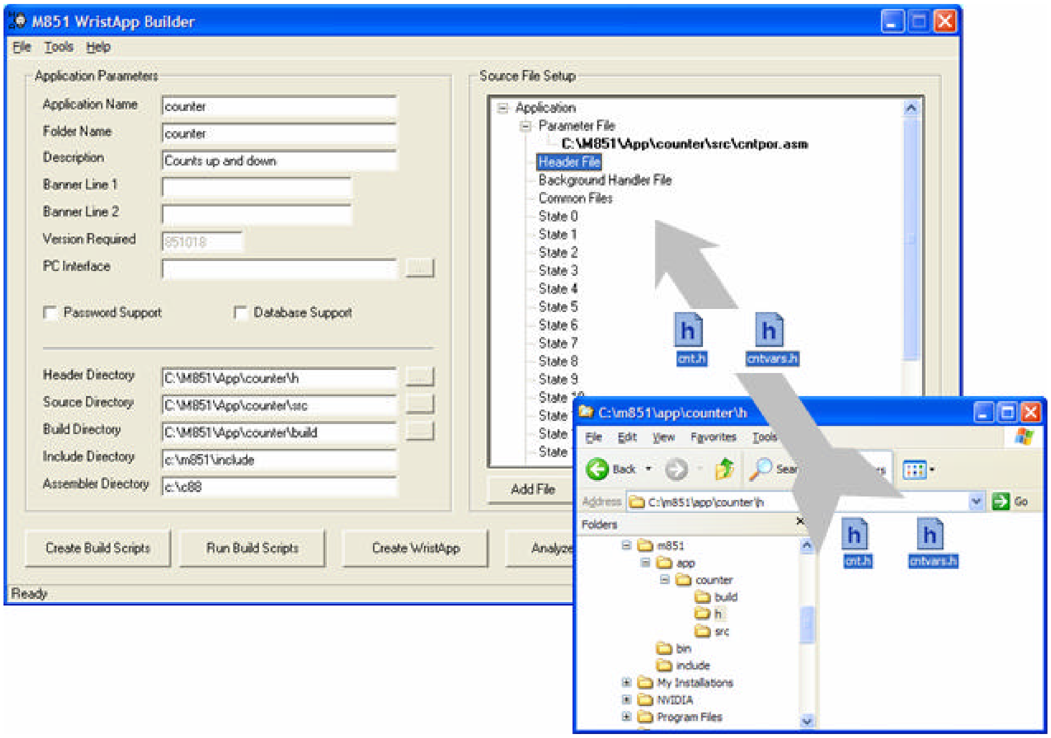


Application Name: Counter WristApp
Folder Name: counter
Description: Counts up and down
Banner Line 1:
Banner Line 2:
Version Required: 851018
PC Interface:
□ Password Support □ Database Support
Header Directory: C:\M851\App\counter\h
Source Directory: C:\M851\App\counter\src
Build Directory: C:\M851\App\counter\build
Include Directory: c:\m851\include
Assembler Directory: c:\c88

```
⊟ Application
   ⊟ Parameter File
        C:\M851\App\counter\src\cntpor.asm
     Header File
     Background Handler File
     Common Files
     State 0
     State 1
     State 2
     State 3
     State 4
     State 5
     State 6
     State 7
     State 8
     State 9
     State 10
     State 11
     State 12
     State 13
```

Ready 8/16/2002 3:44 PM

Open File Explorer and select the files to be added. Then click on the highlighted files and drag them over the Source File Setup List window.



Application Name: counter
Folder Name: counter
Description: Counts up and down
Banner Line 1:
Banner Line 2:
Version Required: 851018
PC Interface:
□ Password Support □ Database Support
Header Directory: C:\M851\App\counter\h
Source Directory: C:\M851\App\counter\src
Build Directory: C:\M851\App\counter\build
Include Directory: c:\m851\include
Assembler Directory: c:\c88

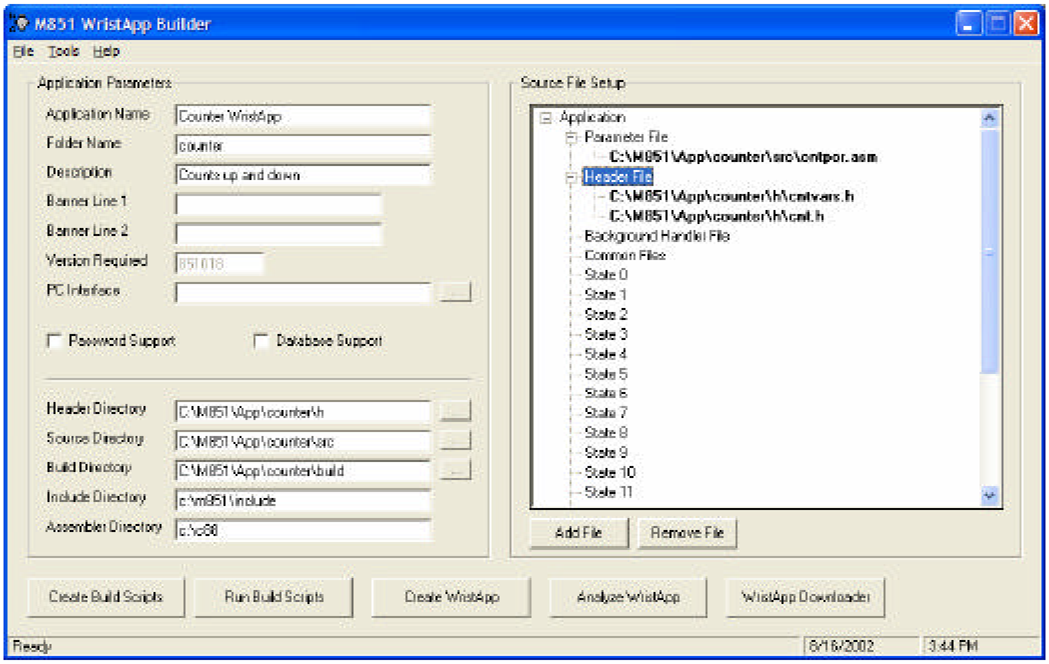
```
⊟ Application
   ⊟ Parameter File
        C:\M851\App\counter\src\cntpor.asm
     Header File
     Background Handler File
     Common Files
     State 0
     State 1
     State 2
     State 3
     State 4
     State 5
     State 6
     State 7
     State 8
     State 9
     State 10
     State
     State
     State
```

Ready

Address: C:\m851\app\counter\h

```
⊟ m851
   ⊟ app
      ⊟ counter
           build
           h
           src
     bin
     include
⊞ My Installations
⊞ NVIDIA
⊞ Program Files
```

After this operation, the files CNT.H and CNTVARS.H will be added under the “Header File” section. See figure below.

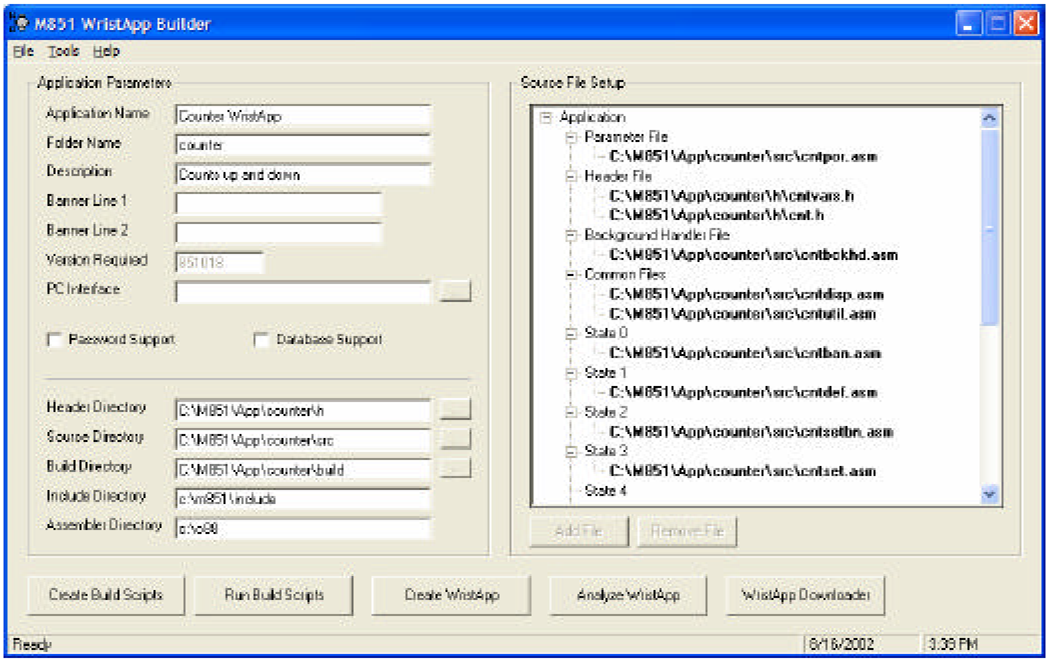


Application Name: Counter WristApp
Folder Name: counter
Description: Counts up and down
Banner Line 1:
Banner Line 2:
Version Required: 851018
PC Interface:
☐ Password Support ☐ Database Support
Header Directory: C:\M851\App\counter\h
Source Directory: C:\M851\App\counter\src
Build Directory: C:\M851\App\counter\build
Include Directory: c:\m851\include
Assembler Directory: c:\ic88

```
Application
  Parameter File
    C:\M851\App\counter\src\cntpor.asm
  Header File
    C:\M851\App\counter\h\cntvars.h
    C:\M851\App\counter\h\cnt.h
  Background Handler File
  Common Files
  State 0
  State 1
  State 2
  State 3
  State 4
  State 5
  State 6
  State 7
  State 8
  State 9
  State 10
  State 11
```

Ready | 8/16/2002 | 3:44 PM

The figure below shows all the files added into their respective sections for the counter wristapp.



Application Name: Counter WristApp
Folder Name: counter
Description: Counts up and down
Banner Line 1:
Banner Line 2:
Version Required: 851018
PC Interface:
☐ Password Support ☐ Database Support
Header Directory: C:\M851\App\counter\h
Source Directory: C:\M851\App\counter\src
Build Directory: C:\M851\App\counter\build
Include Directory: c:\m851\include
Assembler Directory: c:\ic88

```
Application
  Parameter File
    C:\M851\App\counter\src\cntpor.asm
  Header File
    C:\M851\App\counter\h\cntvars.h
    C:\M851\App\counter\h\cnt.h
  Background Handler File
    C:\M851\App\counter\src\cntbckhd.asm
  Common Files
    C:\M851\App\counter\src\cntdisp.asm
    C:\M851\App\counter\src\cntutil.asm
  State 0
    C:\M851\App\counter\src\cntban.asm
  State 1
    C:\M851\App\counter\src\cntdef.asm
  State 2
    C:\M851\App\counter\src\cntsetbn.asm
  State 3
    C:\M851\App\counter\src\cntset.asm
  State 4
```

Ready | 8/16/2002 | 3:39 PM

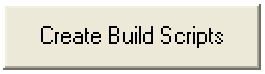
2.11.3 Saving the Current Workspace

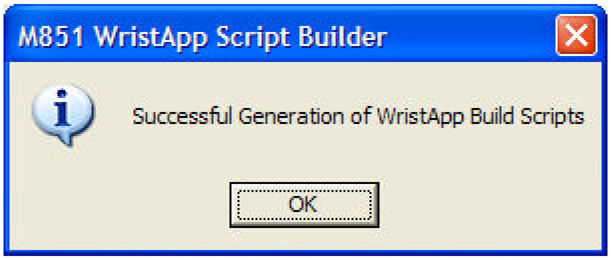
Selecting **File\Save** menu option will store the current workspace under the filename `C:\M851\APP\`*`appname`*`\build\appname.scr`. It can be loaded again by using the **File\Open** menu option.

2.11.4 Creating the Build Scripts

Clicking on the “Create Build Scripts” button will create all the required scripts that automates the assembly and linking of the source files. All script files will be created under the

`C:\M851\APP\appname\BUILD` directory. This process will also save the current workspace under the filename `C:\m851\app\appname\build\appname.scr`.



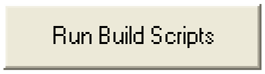


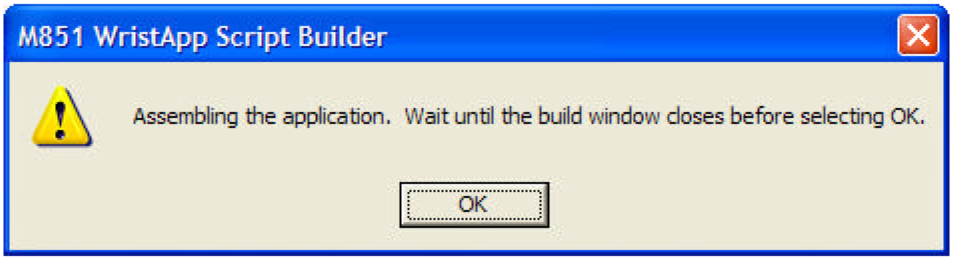
Successful Generation of WristApp Build Scripts

Once the build scripts are created, it is not required to create them again during the debugging process.

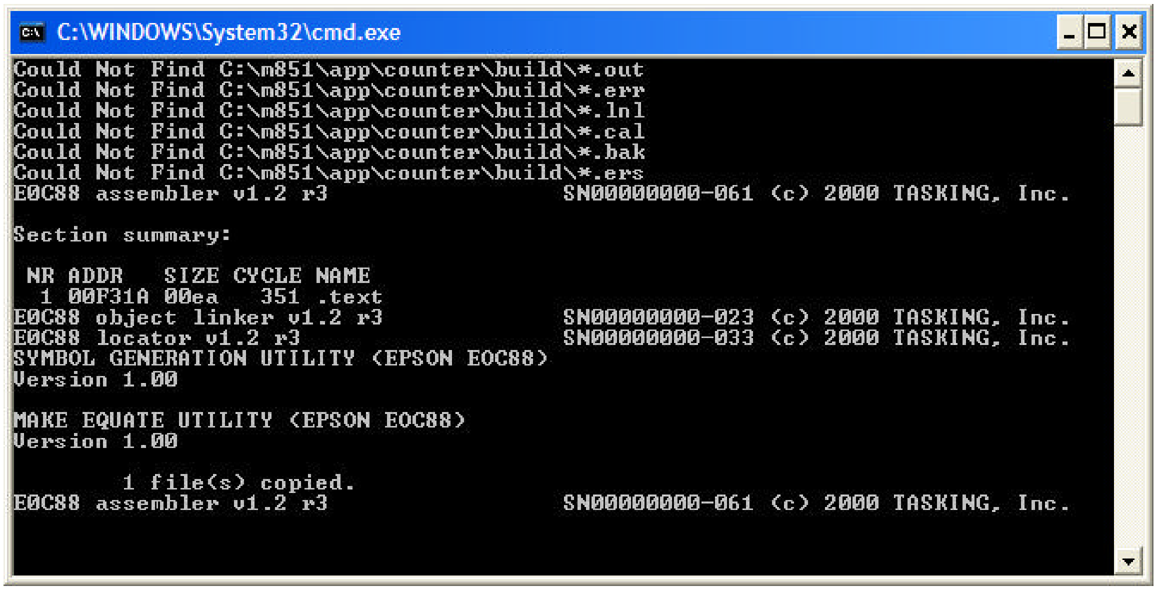
2.11.5 Executing the Build Scripts

Clicking on the “Run Build Scripts” button will execute all the scripts generated in the previous section. This process will open up a command window where all the required scripts are executed. The build process will take some time to complete.





Assembling the application. Wait until the build window closes before selecting OK.



```
Could Not Find C:\m851\app\counter\build\*.out
Could Not Find C:\m851\app\counter\build\*.err
Could Not Find C:\m851\app\counter\build\*.lnl
Could Not Find C:\m851\app\counter\build\*.cal
Could Not Find C:\m851\app\counter\build\*.bak
Could Not Find C:\m851\app\counter\build\*.ers
E0C88 assembler v1.2 r3                SN00000000-061 (c) 2000 TASKING, Inc.

Section summary:

 NR ADDR   SIZE CYCLE NAME
  1 00F31A 00ea    351 .text
E0C88 object linker v1.2 r3            SN00000000-023 (c) 2000 TASKING, Inc.
E0C88 locator v1.2 r3                  SN00000000-033 (c) 2000 TASKING, Inc.
SYMBOL GENERATION UTILITY (EPSON EOC88)
Version 1.00

MAKE EQUATE UTILITY (EPSON EOC88)
Version 1.00

        1 file(s) copied.
E0C88 assembler v1.2 r3                SN00000000-061 (c) 2000 TASKING, Inc.
```

Build Window

A successful build of the code sections for the counter will generate the following SRE files:

- COMMON.SRE
- STATE0.SRE
- STATE1.SRE
- STATE2.SRE
- STATE3.SRE

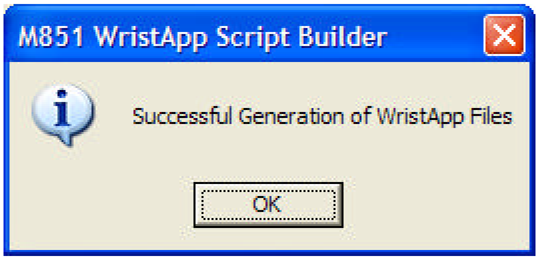
⚠ **NOTE:** Wait until the build process is complete. Do not click on the “Create WristApp” button until the command window is closed.

⚠ **WARNING:** Executing the build scripts does not nescessarily mean that all the code sections has been compiled properly.

2.11.6 Creating the WristApp Downloadable Files

Clicking on the “Create WristApp” button will create the files that are downloaded to the watch.

If all the code sections has been compiled properly with no compile and build errors, the distribution files are generated for download and testing.



Successful Generation of WristApp Files

The distribution files are described below:

File	Description
appname.**app**	*This file is required by the PIM. This contains information about the application such as: user mode banner names, the code file, the parameter file, password support, firmware version requirements and PC WristApp Interface file.* *The* ***appname*** *is the name of wristapp.*
appname.**txt**	*Description file for the PIM. This is a template only. Modify this template and save it under another directory for distribution*
*appname*_**par**_*nnn*.**bin**	*The parameter file contains information required by the watch that determines how the watch behaves in the system and its resource requirements.* ***appname*** *is the name of wristapp.* ***nnn*** *is the version number of the required M851 firmware.*

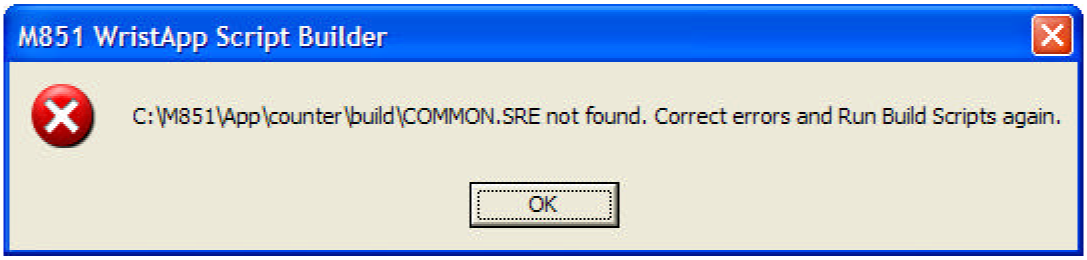
*appname*_**code**_*nnn*.**bin**	*This is the WristApp code stored in a format that the watch can readily grab the correct section to be loaded into the overlay area for execution.* *The* **appname** *is the name of wristapp.* **nnn** *is the version number of the required M851 firmware.*

For the counter wristapp, these are the following files generated:

- counter.app
- counter.txt
- counter_par_018.bin
- counter_code_018.bin

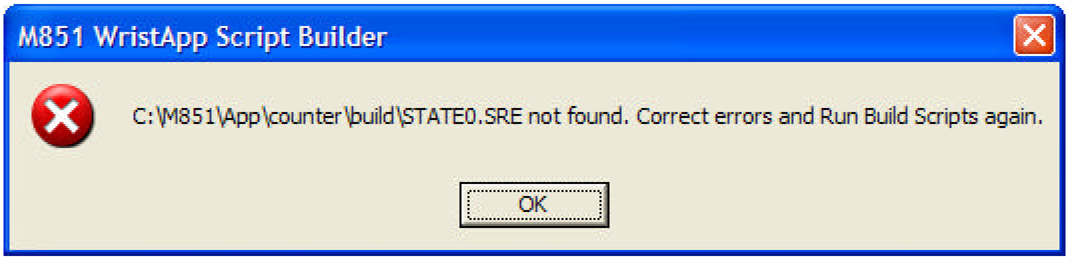
If there are no errors in the source files, all the required files to build the downloadable file will be available and executing the Create WristApp Downloadable Files would be completed.

If the Create WristApp button displays a message indicating that a ?????????.SRE is not found (as shown in the screen snapshots below), this indicates that the build script was unable to complete compiling the section due to errors in the source files attached to a section.



C:\M851\App\counter\build\COMMON.SRE not found. Correct errors and Run Build Scripts again.

Source files attached to the COMMON section have errors.



C:\M851\App\counter\build\STATE0.SRE not found. Correct errors and Run Build Scripts again.

Source files attached to the STATE0 section have erro rs.

If an error exists then you can view the source of the errors by opening the following files:

File	**Description**
sourcename.**ers**	*This error file is generated by the assembler (AS88.EXE). If successful, the output of the assembler is an OBJ file.* *The* ***sourcename*** *could be the section that generated the error. For example: common.ers, state0.ers, state1.ers or param.ers.*
sourcename.**elk**	*This error file is generated by the linker (LK88.EXE). If successful, the output of the linker is an OUT file.*

*The **sourcename** could be the section that generated the error. For example: common.elk, state0.elk, state1.elk or param.elk.*

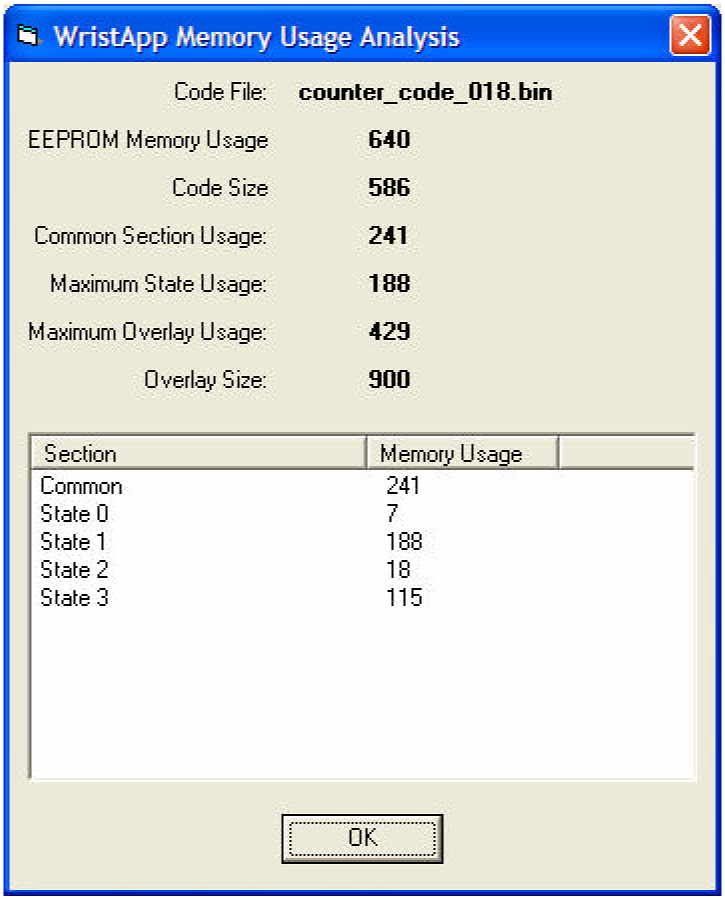
sourcename.**elc**

This error file is generated by the locator (LC88.EXE). If successful, the output of the locator is an SRE file.

*The **sourcename** could be the section that generated the error. For example: common.elc, state0.elc, state1.elc or param.elc.*

2.11.7 WristApp Memory Usage Analysis

Clicking on the "Analyze WristApp" button will open up a window that shows the memory usage of the wristapp and determines if it can fit in the overlay memory area of the M851. A sample display is shown below. The maximum overlay usage must be within the 900 byte limitation.



Code File: **counter_code_018.bin**

EEPROM Memory Usage **640**

Code Size **586**

Common Section Usage: **241**

Maximum State Usage: **188**

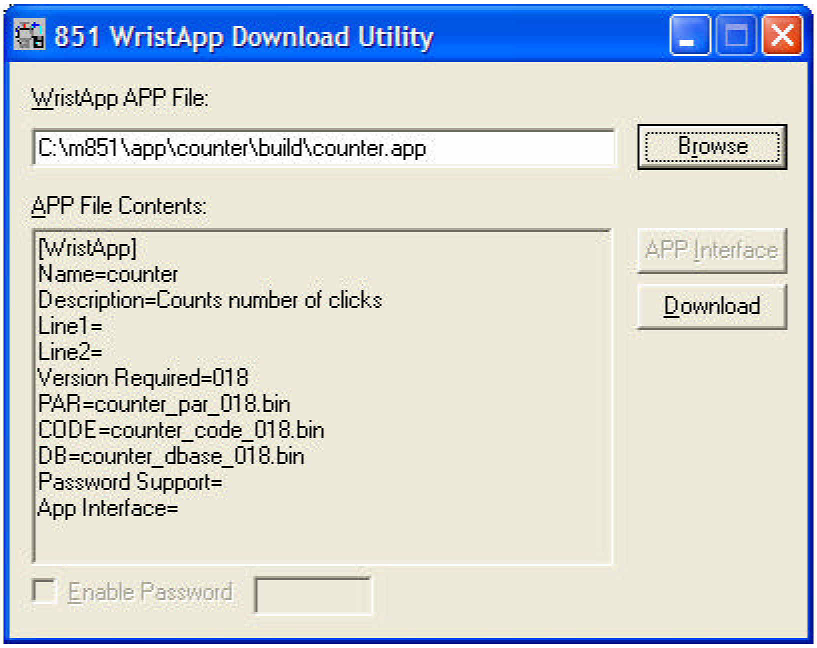
Maximum Overlay Usage: **429**

Overlay Size: **900**

Section	Memory Usage
Common	241
State 0	7
State 1	188
State 2	18
State 3	115

2.11.8 Downloading and Testing the WristApp

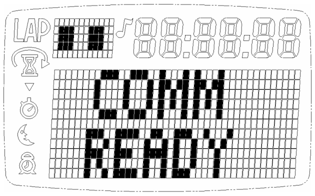
Clicking on the "WristApp Downloader" button will execute the "**M851 WristApp Download Utility**". Once open, click on the "**Browse**" button and select the *appname*.**app** indicated in the previous section.



C:\m851\app\counter\build\counter.app

```
[WristApp]
Name=counter
Description=Counts number of clicks
Line1=
Line2=
Version Required=018
PAR=counter_par_018.bin
CODE=counter_code_018.bin
DB=counter_dbase_018.bin
Password Support=
App Interface=
```

Connect the watch to the PC using the USB cable. Once the watch displays "COMM READY", click on the "**Download**" button of the utility.



NOTE: The M851 WristApp Download Utility can be executed directly. It is located in the C:\M851\BIN directory.

2.11.9 Creating a Description File

Prepare a description file that will be used by the PIM to describe the wristapp. The filename is the same as the app file name. In this example, the description file is: COUNTER.TXT. The text below shows a sample entry for the description file.

```
WRISTAPP: COUNTER


Description:
------------

 The wristapp simulates a mechanical counter.  The user can select either a
 count up or count down operation.
```

```
Usage:
------

 Default State:
 --------------

  The arrow in the upper dot matrix region indicates the operation of the
  wristapp.  Arrow-Up indicates a count-up.  Arrow-Down indicates a count
  down operation.

  The digit in the main dot matrix region indicates the current count.

  Switches:

   MODE        - proceed to the next mode or primary time zone
   START/SPLIT - increment or decrement the count depending on direction
   STOP/RESET  - hold to reset the counter
   CROWN-SET   - pull crown to set to set the counter start value and direction


 Set State:
 ----------

  There are two fields that can be set in this setting operation:

   (1) counter start value;
   (2) counter direction

  Switches:

   MODE         - proceed to the next setting field position with wrap around
   STOP/RESET   - proceed to the previous setting field position with wrap around
   CROWN-HOME   - push crown to home to complete setting operation
   CROWN-CW/CCW - change value of the current setting field.


Files:
------

 counter.app          - application info
 counter.txt          - application description (this file)
 counter_par_018.bin  - application initialization parameter list
 counter_code_018.bin - application code
```

2.11.10 Distributing the WristApp

The following files generated by the system and one manually created by the user will be used for distribution of the wristapp.

	Filename	Description
	application_name.APP	*Information file required by PIM.*
	application_name.TXT	*Description of the wristapp and its operation.*
	*application_name*_PAR_018.BIN	*Parameter file required by M851 OS to initialize the wristapp in the system.*
	*application_name*_CODE_018.BIN	*WristApp code.*
	*application_name*_DBASE_018.BIN	*WristApp database file.*
	application_name.DLL	*WristApp PC interface*

The counter wristapp distribution files:

	Filename	**Description**
	COUNTER.APP	*Information file required by PIM.*
	COUNTER.TXT	*Description of the wristapp and its operation.*
	COUNTER_PAR_018.BIN	*Parameter file required by M851 OS to initialize the wristapp in the system.*
	COUNTER_CODE_018.BIN	*Counter WristApp code.*

3 Trademarks

TIMEX is a registered trademark and service mark of Timex Corporation.
TIMEX DATA LINK and WristApp are trademarks of Timex Corporation in the U.S. and other countries.
Night-Mode is a registered trademark of Timex Corporation.
INDIGLO is a registered trademark of Indiglo Corporation.