Z-Stack
Compile Options

Document Number: SWRA188

Texas Instruments, Inc.
San Diego, California USA

Version	Description	Date
1.0	Initial release.	04/23/2008
1.1	Updated for 2.1.0 release.	04/29/2008
1.2	Updated for 2.2.0 release	03/30/2009
1.3	Added ZDO_NV_SAVE_RFDs, ZDAPP_UPDATE_NWK_NV_TIME, & MT_SYS_OSAL_NV_READ_CERTIFICATE_DATA.	02/04/2010

Table of Contents

1. Introduction

1.1. Scope

This document provides information and procedures for using compiler options with Texas Instruments Z-Stack™, and it's recommend that you don't change compile flags that aren't listed in this document.

2. Requirements

2.1. Target Development System Requirements

Z-Stack is built on top of the IAR Embedded Workbench suite of software development tools (www.iar.com). These tools support project management, compiling, assembling, linking, downloading, and debugging for various development platforms. The following are required support for the Z-Stack target development system:

Platform/Target	Compiler/Tool	Compiler Version
SmartRF05EB+CC2530	IAR EW8051	7.51
EXP5438+CC2520	IAR EW430	4.20.1
MSP2618+CC2520	IAR EW430	4.20.1

3. Using Z-Stack Compile Options

3.1. Selecting the Logical Device Type

ZigBee devices can be configured in one of three ways (each of these device types are explained in the ZStack Developer's Guide), and your application will be hosted on one (or more) of these device types:

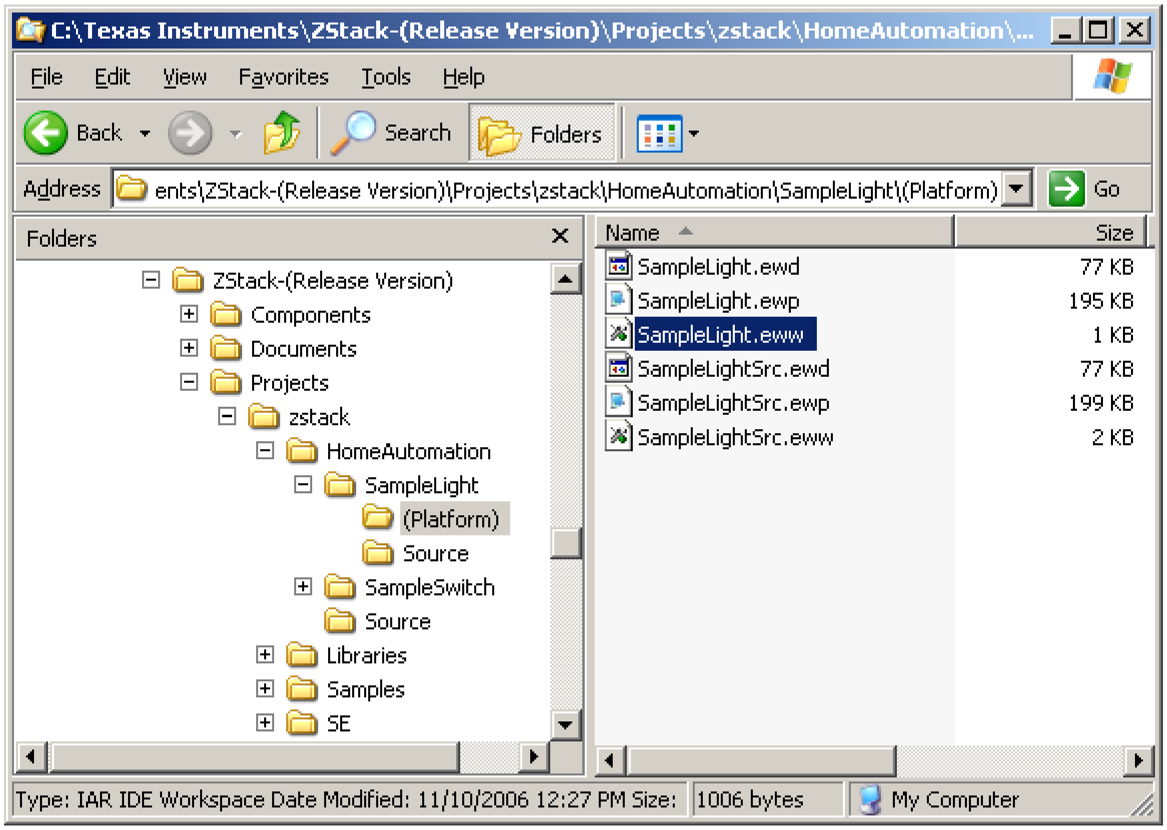
- ZigBee Coordinator – This device is configured to start the IEEE 802.15.4 network and will serve as the PAN Coordinator in that network.
- ZigBee Router – This device is configured to associate with a ZigBee Coordinator, then allow other routers or end devices to associate with it. It will route data packets in the network.
- ZigBee End Device – This device is configured to join a pre-existing network and will associate with a ZigBee Coordinator or ZigBee Router.

3.2. Locating Compile Options

Compile options for a specific project are located in two places. Options that are rarely, if ever, changed are located in linker control files, one for each logical device type discussed above. User-defined options and ones that change to enable/disable features are located in the IAR project file. For demonstration purposes, these two files for the SampleLight Coordinator project will be examined. Access to all other Z-Stack projects will be similar.

3.2.1. Compile Options In Linker Control Files

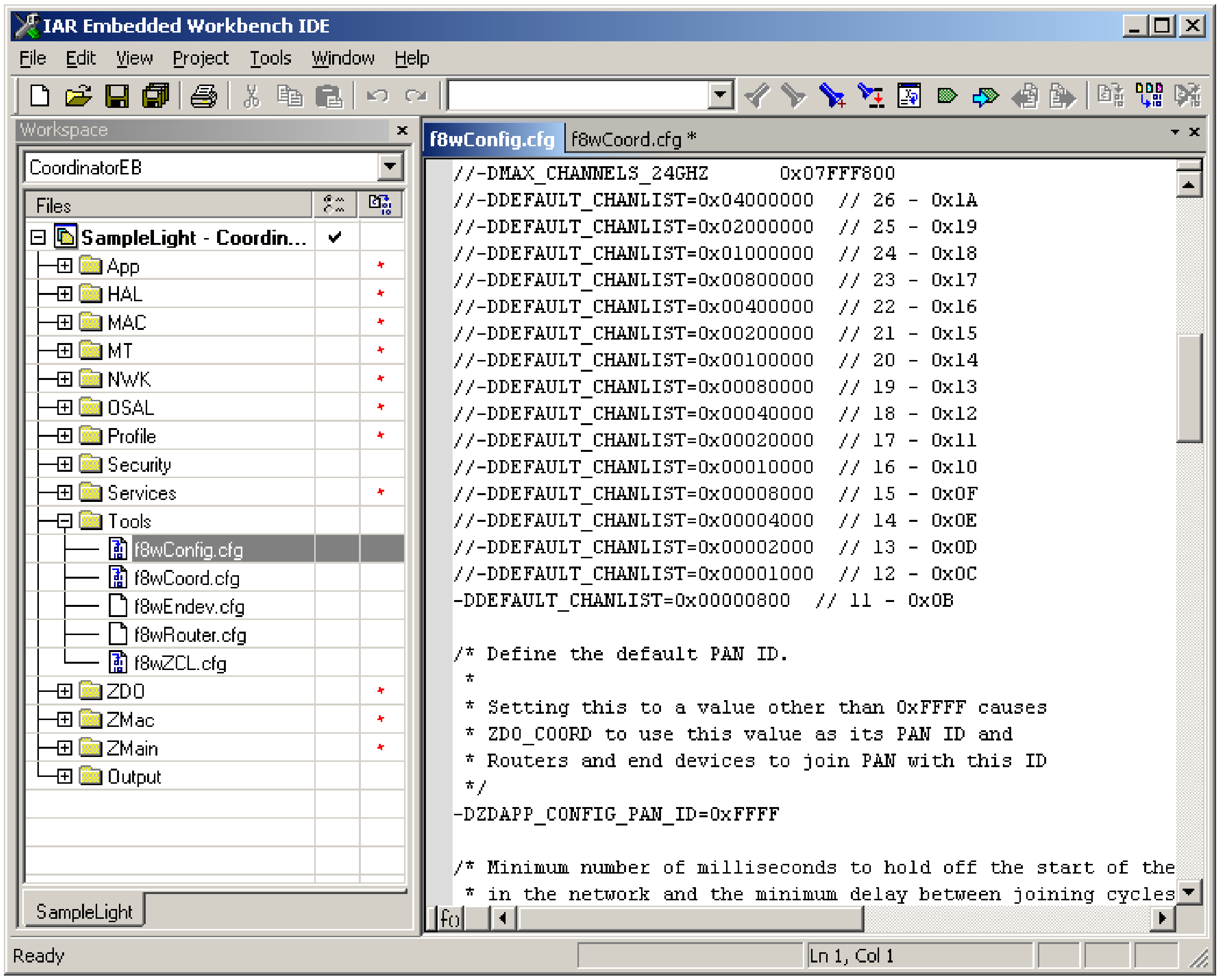
SampleLight project files are found in the **..\Projects\zstack\HomeAutomation\SampleLight\(Platform)** folder:



Name	Size
SampleLight.ewd	77 KB
SampleLight.ewp	195 KB
SampleLight.eww	1 KB
SampleLightSrc.ewd	77 KB
SampleLightSrc.ewp	199 KB
SampleLightSrc.eww	2 KB

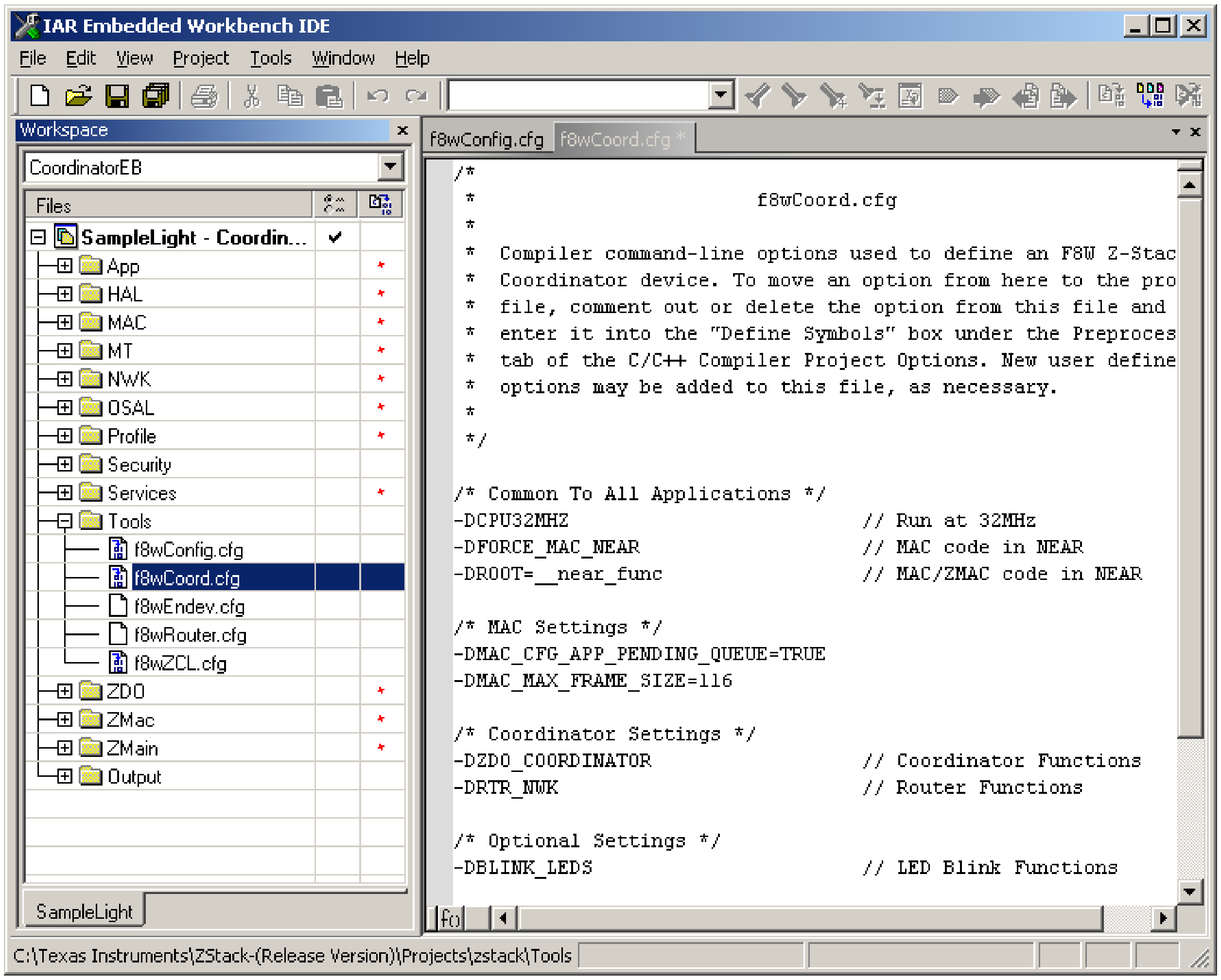
Type: IAR IDE Workspace Date Modified: 11/10/2006 12:27 PM Size: 1006 bytes

Open the project by double-clicking on the *SampleLight.eww* file, select the *Coordinator* configuration from the pull-down list below **Workspace**, and then open the **Tools** folder. Several linker control files are located in the **Tools** folder. This folder contains various configuration files and executable tools used in Z-Stack projects. Generic compile options are defined in the *f8wConfig.cfg* file. This file, for example, specifies the channel(s) and the PAN ID that will be used when a device starts up. This is the recommended location for a user to establish specific channel settings for their projects. This allows developers set up "personal" channels to avoid conflict with others. Device specific compile options are located in the *f8wCoord.cfg*, *f8wEndev.cfg*, and *f8wRouter.cfg* files:



```
//-DMAX_CHANNELS_24GHZ      0x07FFF800
//-DDEFAULT_CHANLIST=0x04000000  // 26 - 0x1A
//-DDEFAULT_CHANLIST=0x02000000  // 25 - 0x19
//-DDEFAULT_CHANLIST=0x01000000  // 24 - 0x18
//-DDEFAULT_CHANLIST=0x00800000  // 23 - 0x17
//-DDEFAULT_CHANLIST=0x00400000  // 22 - 0x16
//-DDEFAULT_CHANLIST=0x00200000  // 21 - 0x15
//-DDEFAULT_CHANLIST=0x00100000  // 20 - 0x14
//-DDEFAULT_CHANLIST=0x00080000  // 19 - 0x13
//-DDEFAULT_CHANLIST=0x00040000  // 18 - 0x12
//-DDEFAULT_CHANLIST=0x00020000  // 17 - 0x11
//-DDEFAULT_CHANLIST=0x00010000  // 16 - 0x10
//-DDEFAULT_CHANLIST=0x00008000  // 15 - 0x0F
//-DDEFAULT_CHANLIST=0x00004000  // 14 - 0x0E
//-DDEFAULT_CHANLIST=0x00002000  // 13 - 0x0D
//-DDEFAULT_CHANLIST=0x00001000  // 12 - 0x0C
-DDEFAULT_CHANLIST=0x00000800  // 11 - 0x0B

/* Define the default PAN ID.
 *
 * Setting this to a value other than 0xFFFF causes
 * ZDO_COORD to use this value as its PAN ID and
 * Routers and end devices to join PAN with this ID
 */
-DZDAPP_CONFIG_PAN_ID=0xFFFF

/* Minimum number of milliseconds to hold off the start of the
 * in the network and the minimum delay between joining cycles
```

The SampleLight Coordinator project uses the *f8wCoord.cfg* file. As shown below, compile options that are specific to Coordinator devices and options that provide "generic" Z-Stack functions are included in this file:



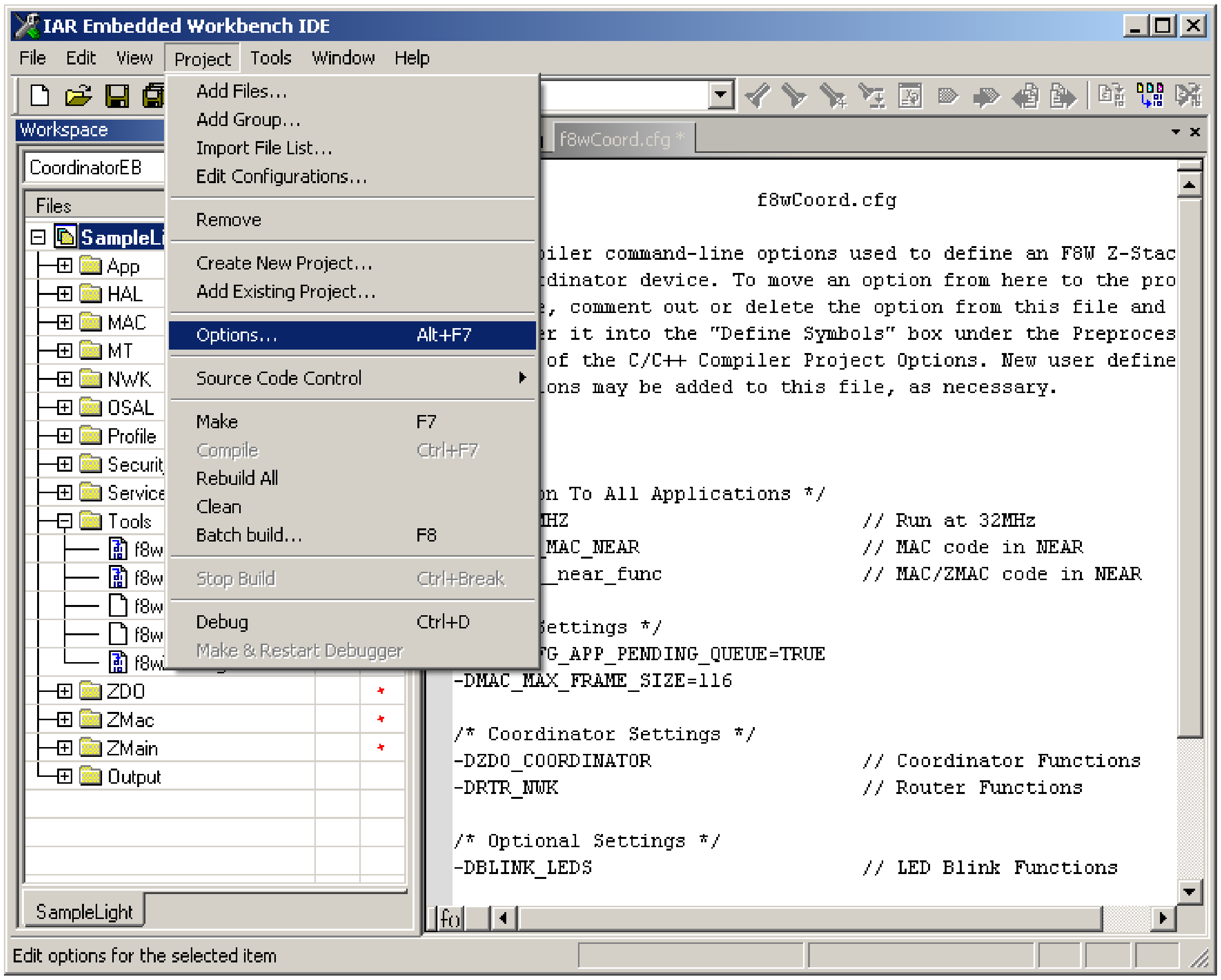
```
/*
 *                     f8wCoord.cfg
 *
 *  Compiler command-line options used to define an F8W Z-Stac
 *  Coordinator device. To move an option from here to the pro
 *  file, comment out or delete the option from this file and
 *  enter it into the "Define Symbols" box under the Preproces
 *  tab of the C/C++ Compiler Project Options. New user define
 *  options may be added to this file, as necessary.
 *
 */

/* Common To All Applications */
-DCPU32MHZ                          // Run at 32MHz
-DFORCE_MAC_NEAR                    // MAC code in NEAR
-DROOT=__near_func                  // MAC/ZMAC code in NEAR

/* MAC Settings */
-DMAC_CFG_APP_PENDING_QUEUE=TRUE
-DMAC_MAX_FRAME_SIZE=116

/* Coordinator Settings */
-DZDO_COORDINATOR                   // Coordinator Functions
-DRTR_NWK                           // Router Functions

/* Optional Settings */
-DBLINK_LEDS                        // LED Blink Functions
```

The *f8wCoord.cfg* file is used by all projects that build Coordinator devices. Therefore, any change made to this file will affect all Coordinators. In a similar manner, the *f8wRouter.cfg* and *f8wEnd.cfg* files affect all Router and End-Device projects, respectively.

To add a compile option to all projects of a certain device type, simply add a new line to the appropriate linker control file. To disable a compile option, comment that option out by placing // at the left edge of the line. You could also delete the line but this is not recommended since the option might need to be re-enabled at a later time.

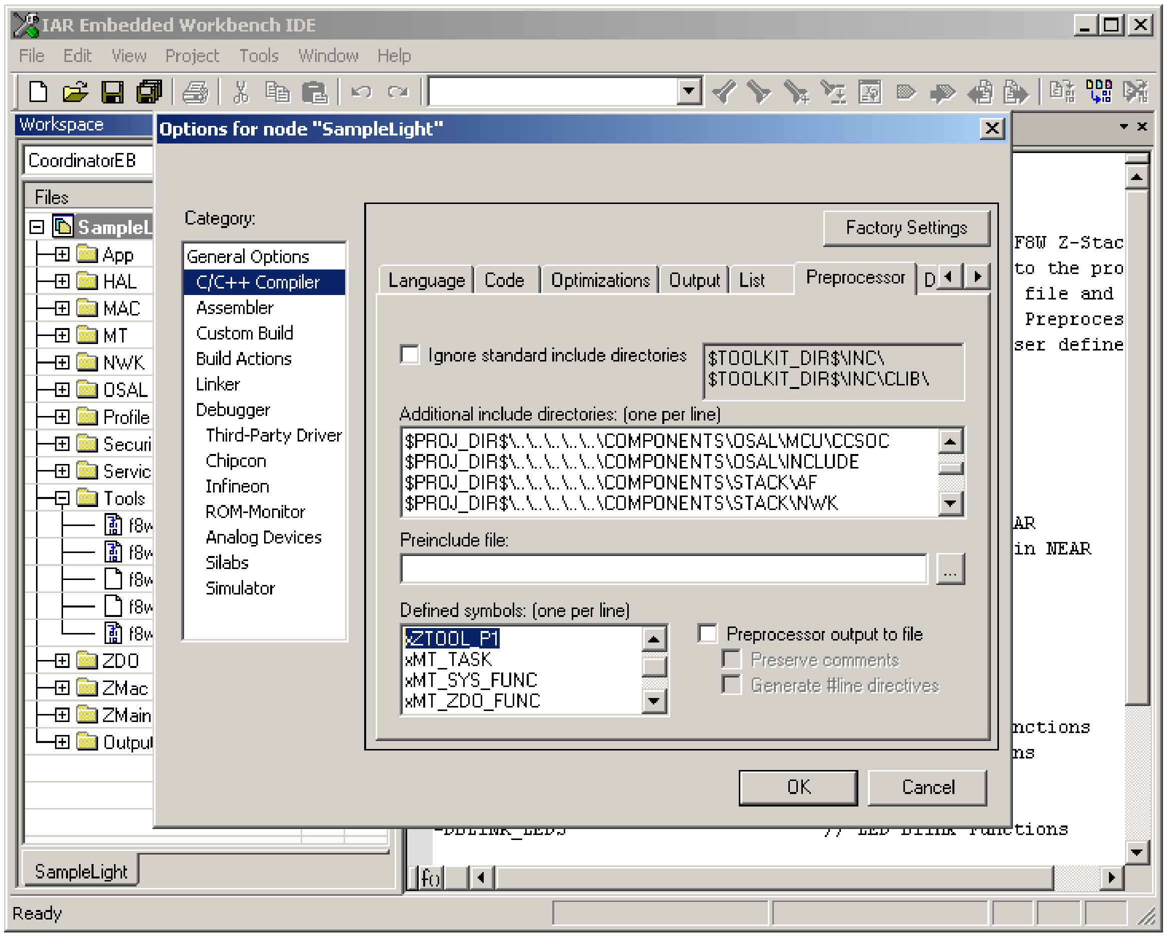
3.2.2. Compile Options In IAR Project Files

The compile options for each of the supported configurations are stored in the *SampleLight.ewp* file. To modify these compile options, first select **SampleLight – Coordinator**. Then select the **Options…** item from the **Project** pull-down menu:



```
                         f8wCoord.cfg

piler command-line options used to define an F8W Z-Stac
rdinator device. To move an option from here to the pro
e, comment out or delete the option from this file and
er it into the "Define Symbols" box under the Preproces
 of the C/C++ Compiler Project Options. New user define
ions may be added to this file, as necessary.




on To All Applications */
MHZ                                  // Run at 32MHz
_MAC_NEAR                            // MAC code in NEAR
_near_func                           // MAC/ZMAC code in NEAR

Settings */
FG_APP_PENDING_QUEUE=TRUE
-DMAC_MAX_FRAME_SIZE=116

/* Coordinator Settings */
-DZDO_COORDINATOR                    // Coordinator Functions
-DRTR_NWK                            // Router Functions

/* Optional Settings */
-DBLINK_LEDS                         // LED Blink Functions
```

Select the **C/C++ Compiler** item and click on the **Preprocessor** tab. The compile options for this configuration are located in the box labeled *Defined symbols: (one per line)*:



Ignore standard include directories
```
$TOOLKIT_DIR$\INC\
$TOOLKIT_DIR$\INC\CLIB\
```
Additional include directories: (one per line)
```
$PROJ_DIR$\..\..\..\..\..\COMPONENTS\OSAL\MCU\CCSOC
$PROJ_DIR$\..\..\..\..\..\COMPONENTS\OSAL\INCLUDE
$PROJ_DIR$\..\..\..\..\..\COMPONENTS\STACK\AF
$PROJ_DIR$\..\..\..\..\..\COMPONENTS\STACK\NWK
```
Preinclude file:

Defined symbols: (one per line)
```
xZTOOL_P1
xMT_TASK
xMT_SYS_FUNC
xMT_ZDO_FUNC
```
Preprocessor output to file
Preserve comments
Generate #line directives

To add a compile option to this configuration, simply add the item on a new line within this box. To disable a compile option, place an x at the left edge of the line. Note that the **ZTOOL_P1** option has been disabled in the example shown above. This option could have been deleted but this is not recommended since it might need to be re-enabled at a later time.

3.3. Using Compile Options

Compile options are used to select features that are provided in the source files. Most compile options act as on/off switches for specific sections within source programs. Some options are used to provide a user-defined numerical value, such as *DEFAULT_CHANLIST*, to the compiler to override default values.

Each of the Z-Stack sample applications (ex. SampleApp) provide an IAR project file which specifies the compile options to be used for that specific project. The programmer can add or remove options as needed to include or exclude portions of the available software functions. Note that changing compile options may require other changes to the project file (see 3.2). For example, adding the *MT_NWK* options requires *MT_NWK.c* to be in the list of source files and the use of the appropriate MT-enabled network library - if you are changing the SampleLight Coordinator project, which normally uses the *Router<Platform>.lib* file, the *Router<Platform>MT.lib* file must be used instead.

The next sections of this document provide lists of the supported compile options with a brief description of what feature they enable or disable. Options that are listed as "*do not change*" are required for proper operation of the compiled programs. Options that are listed as *"do not use"* are not appropriate for use with the board.

4. Supported Compile Options and Definitions

4.1. General Compile Options

Please read the ZStack Developer's Guide before trying to change any of these options, it describes the features that these options change or enable. The compile options in the following table can be changed or set to select desired features, a lot of these compile options are set and described in f8wConfig.cfg.

APS_DEFAULT_INTERFRAME_DELAY	Delay between Tx packets when using fragmentation
APS_DEFAULT_MAXBINDING_TIME	Maximum time in seconds that a Coordinator will wait between receiving match descriptor bind requests to perform binding
APS_DEFAULT_WINDOW_SIZE	Size of a Tx window when using fragmentation
APS_MAX_GROUPS	Maximum number of entries allowed in the groups table
APSC_ACK_WAIT_DURATION_POLLED	Number of 2 milliseconds periods a polling End Device will wait for an APS acknowledgement from the destination device
APSC_MAX_FRAME_RETRIES	Maximum number of retries allowed (at APS layer) after a transmission failure
ASSERT_RESET	Specifies that the device should reset when there's an assertion. When not defined, all LEDs will flash when an assertion occurs.
BEACON_REQUEST_DELAY	Minimum number of milliseconds to delay between each beacon request in a joining cycle
BLINK_LEDS	Enable extended LED blinking functions
DEFAULT_CHANLIST	Change this list in f8wConfig.cfg
EXTENDED_JOINING_RANDOM_MASK	Mask for the random joining delay
HOLD_AUTO_START	Disable automatic start-up of ZDApp event processing loop
LCD_SUPPORTED	Enable LCD emulation – text sent to ZTool serial port
MANAGED_SCAN	Enable delays between channel scans
MAX_BCAST	Maximum number of simultaneous broadcasts supported by a device at any given time
MAX_BINDING_CLUSTER_IDS	Maximum number of cluster IDs in a binding record
MAX_POLL_FAILURE_RETRIES	Number of times retry to poll parent before indicating loss of synchronization with parent. Note that larger value will cause longer delay for the child to rejoin the network
MAX_RREQ_ENTRIES	Number of simultaneous route discoveries in network
MAX_RTG_ENTRIES	Number of entries in the regular routing table plus additional entries for route repair
MAXMEMHEAP	Determines the total memory available for dynamic memory. Every request for an amount of dynamic memory requires dynamic memory space for overhead used in managing the allocated memory. So MAXMEMHEAP does not reflect the total amount of dynamic memory that the user can expect to be usable. As a rule of thumb, each memory allocation requires at least 2+N bytes, where N represents the word-alignment block size of the target CPU (e.g., N=1 on the AVR and CC2430 but N=2 on the MSP430). MAXMEMHEAP must be defined to be less that 32768
NONWK	Disable NWK, APS, and ZDO functionality
NV_INIT	Enable loading of "basic" NV items at device reset
NV_RESTORE	Enables device to save/restore network state information to/from NV
NWK_AUTO_POLL	Enable End Device to poll from the parents automatically
NWK_INDIRECT_MSG_TIMEOUT	Number of milliseconds the parent of a polling End Device will hold a message
NWK_MAX_BINDING_ENTRIES	Maximum number of entries in the binding table
NWK_MAX_DATA_RETRIES	The maximum number of times retry looking for the next hop address of a message
NWK_MAX_DEVICE_LIST	Maximum number of devices in the Association/Device list
NWK_MAX_DEVICES	Maximum number of devices in the network

NWK_START_DELAY	Minimum number of milliseconds to hold off the start of the device in the network and the minimum delay between joining cycles
OSAL_TOTAL_MEM	Track OSAL memory heap usage (display if LCD_SUPPORTED)
POLL_RATE	For end devices only: number of milliseconds to wait between data request polls to its parent. Example POLL_RATE=1000 is a data request every second. This is changed in f8wConfig.cfg.
POWER_SAVING	Enable power saving functions for battery-powered devices
QUEUED_POLL_RATE	This is used after receiving a data indication to poll immediately for queued messages (in milliseconds)
REFLECTOR	Enable binding
REJOIN_POLL_RATE	This is used as an alternate response poll rate only for rejoin request. This rate is determined by the response time of the parent that the device is trying to join
RESPONSE_POLL_RATE	This is used after receiving a data confirmation to poll immediately for response messages (in milliseconds)
ROUTE_EXPIRY_TIME	Number of seconds before an entry expires in the routing table; set to 0 to turn off route expiry
RTR_NWK	Enable Router networking
SECURE	Enable ZigBee security (SECURE=0 to disable, SECURE=1 to enable)
ZAPP_Px	Enable ZApp messages via serial port Px where x is the port (1 or 2)
ZDAPP_CONFIG_PAN_ID	Coordinator’s PAN ID; used by Routers and End Devices to join PAN with this ID
ZDO_COORDINATOR	Enable the device as a Coordinator
ZIGBEEPRO	Enable usage of ZigBee Pro features
ZTOOL_Px	Enable ZTool messages via serial port Px where x is the port (1 or 2)

4.2. Non-changeable Compile Options

These compile options in the following table should not be changed or used. Not all of them are available in every platform:

CC2420DB	Target is a CC2420DB evaluation board **(do not change)**
CC2430BB	Target is a SoC-BB battery board **(do not change)**
CC2430DB	Target is a CC2430DB evaluation board **(do not change)**
CC2430EB	Target is a SmartRF04EB evaluation board **(do not change)**
MSP430FG4618	Target is an MSP430FG4618 processor **(do not change)**
MSP430FG4619	Target is an MSP430FG4619 processor **(do not change)**
MSP430F2618	Target is an MSP430F2618 processor **(do not change)**
CPU6MHZ	Clock rate of the CPU – 6 MHZ **(do not change)**
CPU16MHZ	Clock rate of the CPU – 16 MHZ **(do not change)**
CPU32MHZ	Clock rate of the CPU – 32 MHZ **(do not change)**
MACSIM	Enable MAC simulation **(do not use)**
NWK_TEST	Enable Network test functions **(do not use)**

4.3. Monitor-Test (MT) Compile Options

Please read the Z-Stack Monitor and Test API document before changing any of these compile options. You can enable the following APIs and function associated with the MT_TASK option, but you must include the MT_TASK option.

MT_TASK	Enable Monitor-Test task
MT_AF_FUNC	Enable Monitor-Test processing of AF commands issued from ZTool or ZTrace
MT_AF_CB_FUNC	Enable Monitor-Test processing of AF callbacks registered by ZTool or ZTrace
MT_APP_FUNC	Enable Monitor-Test processing of APP commands issued from ZTool or ZTrace
MT_DEBUG_FUNC	Enable Monitor-Test processing of DEBUG commands issued from ZTool or ZTrace
MT_MAC_FUNC	Enable Monitor-Test processing of MAC commands issued from ZTool or ZTrace
MT_NWK_FUNC	Enable Monitor-Test processing of NWK commands issued from ZTool or ZTrace
MT_NWK_CB_FUNC	Enable Monitor-Test processing of NWK callbacks registered by ZTool or ZTrace
MT_SAPI_FUNC	Enable Monitor-Test processing of SAPI commands issued from ZTool or ZTrace
MT_SAPI_CB_FUNC	Enable Monitor-Test processing of SAPI callbacks registered by ZTool or ZTrace
MT_SYS_FUNC	Enable Monitor-Test processing of SYS commands issued from ZTool or ZTrace

MT_SYS_OSAL_NV_READ_CERTIFICATE_DATA	Default define to FALSE in MT_SYS.c and only applicable if ZCL_KEY_ESTABLISH is defined. If ZCL_KEY_ESTABLISH is defined and MT_SYS_OSAL_NV_READ_CERTIFICATE_DATA is defined to TRUE, then the three NV items containing Certicom certificate data can be read via MT:
	`ZCD_NV_IMPLICIT_CERTIFICATE 0x006A`
	`ZCD_NV_CA_PUBLIC_KEY 0x006B`
	`ZCD_NV_DEVICE_PRIVATE_KEY 0x006C`
	Otherwise, the certificate data cannot be read via MT.

MT_UTIL_FUNC	Enable Monitor-Test processing of UTIL commands issued from ZTool or ZTrace
MT_ZDO_CB_FUNC	Enable Monitor-Test processing of ZDO commands issued from ZTool or ZTrace
MT_ZDO_FUNC	Enable Monitor-Test processing of ZDO commands issued from ZTool or ZTrace
MT_ZDO_MGMT	Enable Monitor-Test processing of ZDO MGMT commands from ZTool or ZTrace

4.4. ZigBee Device Object (ZDO) Compile Options

By default, the mandatory messages (as defined by the ZigBee spec) are enabled in the ZDO. All other message processing is controlled by compile flags. You can enable/disable the options by commenting/uncommenting the compile flags in ZDConfig.h or include/exclude them like other compile flags. There's an easy way to enable all the ZDO Function and Management options: You can use MT_ZDO_FUNC to enable all the ZDO Function options, and MT_ZDO_FUNC and MT_ZDO_MGMT to enable all the ZDO Function plus Management options. Please read the ZStack Developer's Guide and ZStack API document on the use of these messages.

ZDO_NWKADDR_REQUEST	Enable Network Address Request function and response processing
ZDO_IEEEADDR_REQUEST	Enable IEEE Address Request function and response processing
ZDO_MATCH_REQUEST	Enable Match Descriptor Request function and response processing
ZDO_NODEDESC_REQUEST	Enable Node Descriptor Request function and response processing
ZDO_POWERDESC_REQUEST	Enable Power Descriptor Request function and response processing
ZDO_SIMPLEDESC_REQUEST	Enable Simple Descriptor Request function and response processing
ZDO_ACTIVEEP_REQUEST	Enable Active Endpoint Request function and response processing
ZDO_COMPLEXDESC_REQUEST	Enable Complex Descriptor Request function and response processing
ZDO_USERDESC_REQUEST	Enable User Descriptor Request function and response processing
ZDO_USERDESCSET_REQUEST	Enable User Descriptor Set Request function and response processing
ZDO_ENDDEVICEBIND_REQUEST	Enable End Device Bind Request function and response processing
ZDO_BIND_UNBIND_REQUEST	Enable Bind and Unbind Request function and response processing
ZDO_SERVERDISC_REQUEST	Enable Server Discovery Request function and response processing
ZDO_MGMT_NWKDISC_REQUEST	Enable Mgmt Nwk Discovery Request function and response processing
ZDO_MGMT_LQI_REQUEST	Enable Mgmt LQI Request function and response processing
ZDO_MGMT_RTG_REQUEST	Enable Mgmt Routing Table Request function and response processing
ZDO_MGMT_BIND_REQUEST	Enable Mgmt Binding Table Request function and response processing
ZDO_MGMT_LEAVE_REQUEST	Enable Mgmt Leave Request function and response processing
ZDO_MGMT_JOINDIRECT_REQUEST	Enable Mgmt Join Direct Request function and response processing
ZDO_MGMT_PERMIT_JOIN_REQUEST	Enable device to respond to Mgmt Permit Join Request function
ZDO_USERDESC_RESPONSE	Enable device to respond to User Descriptor Request function
ZDO_USERDESCSET_RESPONSE	Enable device to respond to User Descriptor Set Request function
ZDO_SERVERDISC_RESPONSE	Enable device to respond to Server Discovery Request function
ZDO_MGMT_NWKDISC_RESPONSE	Enable device to respond to Mgmt Network Discovery Request function
ZDO_MGMT_LQI_RESPONSE	Enable device to respond to Mgmt LQI Request function
ZDO_MGMT_RTG_RESPONSE	Enable device to respond to Mgmt Routing Table Request function
ZDO_MGMT_BIND_RESPONSE	Enable device to respond to Mgmt Binding Table Request function
ZDO_MGMT_LEAVE_RESPONSE	Enable device to respond to Mgmt Leave Request function
ZDO_MGMT_JOINDIRECT_RESPONSE	Enable device to respond to Mgmt Join Direct Request function
ZDO_MGMT_PERMIT_JOIN_RESPONSE	Enable device to respond to Mgmt Permit Join Request function
ZDO_ENDDEVICE_ANNCE	Enable device to respond to End Device Annce Message function
ZDO_NV_SAVE_RFDs	Default define to TRUE in ZDApp.c and only applicable if NV_RESTORE is defined. If NV_RESTORE is defined and ZDO_NV_SAVE_RFDs is defined to FALSE, then RFD joins will not trigger a call to NLME_UpdateNV() and the delay time between receiving a trigger event and actually invoking NLME_UpdateNV() is extended to the OSAL timer maximum of 65 seconds (see ZDAPP_UPDATE_NWK_NV_TIME). This compile option is intended to be used to greatly extend the life of the NV pages of the RFD's in a network with mobile or purged RFD's. When this flag is defined to FALSE, any RFD children that exist at the time an FFD is reset will not be restored and the FFD can re-issue their network addresses to other joining RFD's.
ZDAPP_UPDATE_NWK_NV_TIME	Default define to 700 msecs and only applicable if NV_RESTORE is defined. The delay time between receiving a network save state trigger event and actually invoking NLME_UpdateNV(). The longer this delay is, the longer the life of the NV pages since this data is very large and in a busy network (especially one with mobile RFD's) the frequency of trigger events could be high.