

Getting Started with InSight Desktop

This document introduces you to InSight Desktop. It highlights basic functionality and shows how to perform some routine tasks. For detailed information about InSight Desktop, refer to its online help or the *InSight Desktop User's Guide*. This document assumes that you have already installed the Developer Kit software, including InSight Desktop.

Contents

Starting InSight Desktop	2
Opening a session file	2
Analyzing session events.....	4
Tracing network events.....	6
Using filters.....	11



Ember Corporation
343 Congress Street
Boston MA 02210
617.951.0200
www.ember.com



wireless semiconductor solutions

Starting InSight Desktop

Start InSight Desktop by double-clicking on the desktop icon or from the Startup menu.

Work areas

The InSight Desktop environment contains the following work areas:

- **Adapters view** lists all adapters and their nodes that are accessible to InSight Desktop over the Ethernet. If no adapters/nodes are accessible, this window is empty.
- **Capture sessions** contain data captured from a node or set of nodes. Each tab represents a different capture session. A capture session is initially live—that is, it contains real-time network data that InSight Desktop captures as it is broadcast. Live capture sessions can be saved to files and replayed at any time.
- **Editor panes** display the data of a capture session. Up to five editor panes display different views of the captured data.

Opening a session file

InSight Desktop is installed with two saved session files, which are accessible from the menu option **Help | Demos**.

Session file preferences

You can set general preferences for InSight Desktop that apply to all capture sessions. InSight Desktop saves these preferences and uses them each time it restarts.

In order to recreate the environment that this document describes, follow these steps:

1. Choose the menu option **File | Preferences**
2. From the Preferences dialog, choose **Decoding**
3. Under Custom Decoders, check **Sensor-Sink Decoder**
4. Click **OK**

Sensor-sink demo

From the Help menu, open the saved session file `Sensor-sink demo`. The InSight Desktop workspace displays this session as follows:

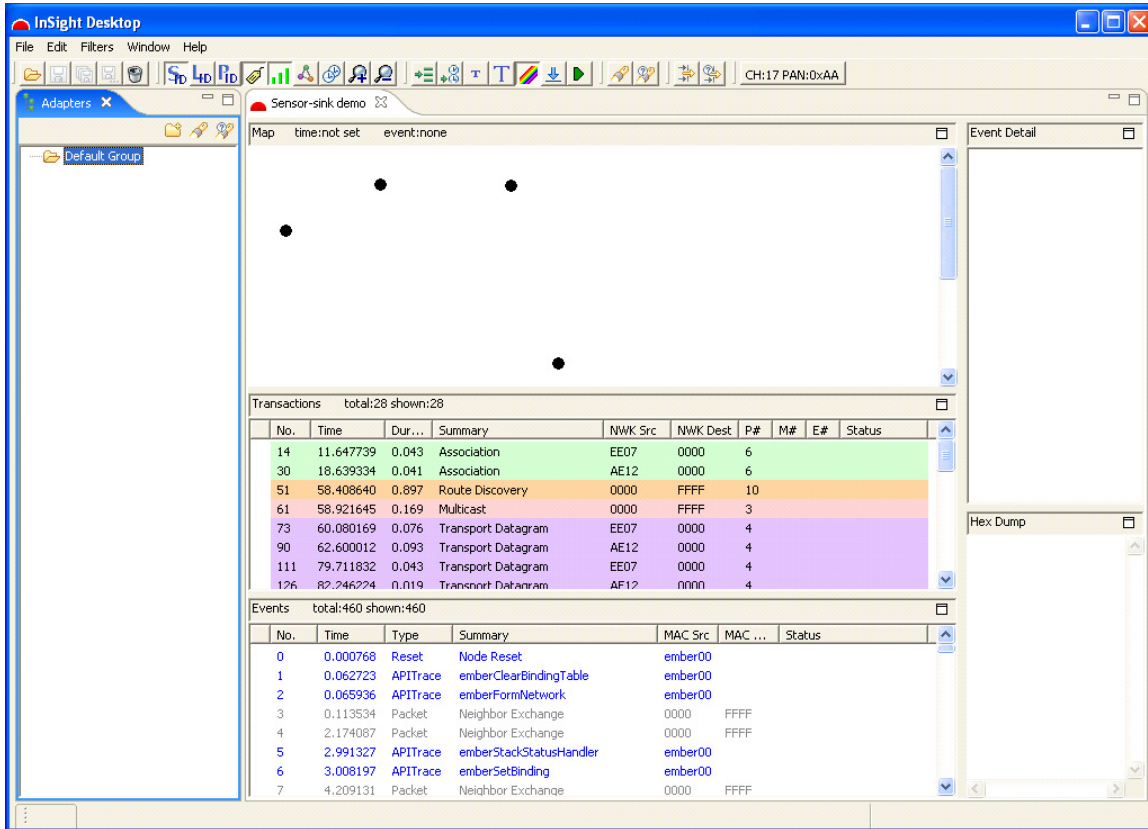


Figure 1. Sensor sink demo

Editor panes

Editor panes display different views of session data. Up to five editor panes can be open at any one time:

- **Map** provides a graphical view of the network, where nodes are displayed as dots with their network identifiers. The map also displays network connectivity and real time traffic: over-the-air messages are represented either as arrows for direct messages, or as circles for broadcasts.
- **Transactions** displays high-level node interactions that comprise multiple events.
- **Events** represents all network occurrences at the most basic level. The Events pane shows individual over-the-air packets, individual calls to Ember's APIs, specific debug events such as node reset, and application-level debug printf statements.
- **Event Detail** displays the decoded contents of the event selected in the Events pane.
- **Hex Dump** displays the data of the selected event in raw bytes. InSight Desktop highlights bytes that map to the data currently selected in the Event Detail pane.

Analyzing session events

You can step through the individual transactions and events that comprise a session, and analyze their data. The following sections focus on two event types:

Over-the-air events

Item 14 in the Transactions pane represents an Association, a ZigBee term that describes a node's attempt to join a network and the network's response:

The screenshot shows the InSight Desktop interface. The top pane is a map showing three nodes: 0x0000 (PAN: 0x01FF, 000D6F000005B037), 0x01FF (PAN: 0x01FF, 000D6F000005B030), and 000D6F000005AE69. A green arrow points from 0x01FF to 0x0000. Below the map is a Transactions table with 28 transactions shown. Transaction 14 is selected, showing an Association from NWK Src EE07 to NWK Dest 0000. Below the transactions is an Events table with 460 events shown. Events 16 through 23 are grouped together, showing packets for Association Request, Ack, Data Request, Ack, Association Response, and Ack.

No.	Time	Dur...	Summary	NWK Src	NWK Dest	P#	M#	E#	Status
14	11.647739	0.043	Association	EE07	0000	6			
30	18.639334	0.041	Association	AE12	0000	6			
51	58.408640	0.897	Route Discovery	0000	FFFF	10			
61	58.921645	0.169	Multicast	0000	FFFF	3			
73	60.080169	0.076	Transport Datagram	EE07	0000	4			
90	62.600012	0.093	Transport Datagram	AE12	0000	4			

No.	Time	Type	Summary	MAC Src	MAC ...	Status
16	11.647...	Packet	Association Request	000D...	0000	
17	11.648...	Packet	802.15.4 Ack	0000	000D...	
19	11.684...	Packet	Data Request	000D...	0000	
20	11.685...	Packet	802.15.4 Ack	0000	000D...	
22	11.688...	Packet	Association Response	000D...	000D...	
23	11.689...	Packet	802.15.4 Ack	000D...	000D...	
24	11.743...	Packet	Neighbor Exchange	EE07	FFFF	

Figure 2. Selection of a transaction and associated display in Map and Events panes

Transactions and associated events

When you click on transaction 14, InSight Desktop updates the data in the Map and Events panes as follows:

- The Map pane displays an arrow from node 0x01FF to node 0x0000. This indicates that node 0x01FF is joining the network through node 0x0000.
- The Events pane shows six over-the-air packets that comprise an Association transaction: 16, 17, 19, 20, 22, and 23. These packets are enclosed by a single bracket to show that they belong to the same transaction.

Event data

Each column in the Events pane provides high level information about a given event:

- **Time** shows the time the event occurred relative to the beginning of the session. If a capture involves multiple nodes, the chronological ordering of events provides a real-time snapshot of interaction between them.

- Summary typically contains the ZigBee term that describes the event type.
- **MAC Src** and **Mac Dest** provide the 802.15.4 MAC layer source and destination addresses, respectively.

Packet details

When you click on transaction 14, InSight Desktop gives focus to event 16, the first event of that transaction. It also displays details of that event in the Event Detail and Hex Dump panes:

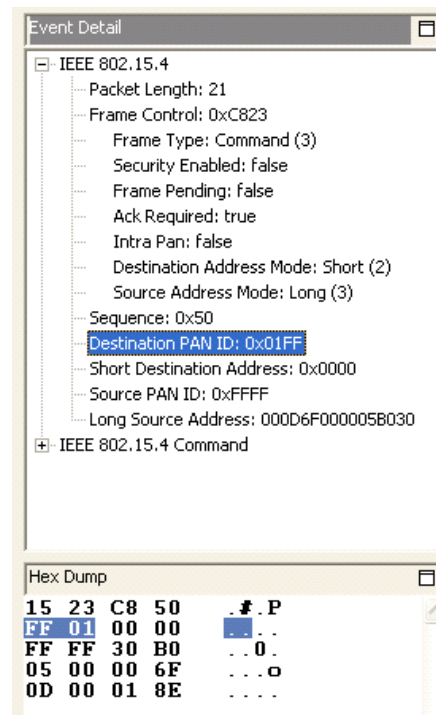


Figure 3. Event Detail and Hex Dump panes

Note: If necessary, resize the Event Detail and Hex Dump panes in order to display all data, by dragging their borders.

The event packet message contains two parts:

- IEEE 802.15.4 layer header
- IEEE 802.15.4 command structure

If you expand the IEEE 802.15.4 Command field (click the + control), InSight Desktop shows more details about this specific command.

- **Command Identifier** shows that this command is type 1, which corresponds to an association request.
- **Capability Info** is set to 0x8E; its subfields specify the properties that pertain to this command and their values.

Hex dump

The Hex Dump pane displays the raw byte-level data of the selected event. When you select any field in the Event Detail pane, the Hex Dump pane highlights the corresponding bytes. For example, when you click on the Destination PAN ID field, the corresponding bytes are highlighted as shown in Figure 3.

Debug events

You can use InSight Desktop to analyze debug events, such as API traces, where processing data is obtained from a running node application that is compiled in debug mode. In contrast to sniffer capture, which can only intercept over-the-air events, InSight Desktop obtains debug events directly from the node itself.

Event 108 of the Sensor-sink demo represents an API Trace event. You can get there by scrolling. Alternatively, you can navigate directly to this line:

5. Choose the menu option Edit | Go To Line
6. Enter 108 and click OK

The Events pane has the following information about event 108:

- The event is bracketed, indicating that it is a self-contained transaction.
- The Type column identifies this event as an API Trace.
- The Summary column identifies the API that was called: `emberPermitJoining()`.
- The MAC Src column reveals that node ember00 made this call.

The Event Detail pane shows the details of the call:

- The API's numeric identifier: 24
- The arguments supplied—in this case, Duration set to 0x00.

Tracing network events

The sensor-sink session file provides a snapshot of a simple sensor network. When this network is in action, three types of related events occur repeatedly:

- Sensor nodes collect data.
- Sensor node transmits data to the sink.
- Receipt of sensor data is acknowledged.

The following sections follow the history of a data packet from its creation on a sensor node, its journey through the EmberZNet stack, transmission over the air, and its eventual receipt by the target node's application. `printf` statements are embedded in the sensor-sink applications in order to facilitate tracing, and are used to verify various events.

Sensor nodes collect data

In the events panel, click on event 109 of the sensor-sink demo. The Event Detail pane, displays this `printf` message:

```
sensor has data ready: 0x7A5F
```

This formatted `printf` message resides in the application layer. When you write an EmberZNet application, it can be useful to insert similar `printf` statements at points of interest. After running the application and capturing its execution through InSight Desktop, you can evaluate these statements in the session file and use their time stamps to determine when they actually executed.

The screenshot displays the InSight Desktop interface for the 'Sensor-sink demo' application. The top section shows a map with three nodes: 0xAE12 (PAN: 0x01FF, 000D6F000005AE69), 0xEE07 (PAN: 0x01FF, 000D6F000005B030), and 0x0000 (PAN: 0x01FF, 000D6F000000). Below the map is a Transactions table with 28 entries. The Events table shows 460 events, with event 109 selected. The Event Detail pane shows a Printf event with the message 'sensor has data ready: 0x7A5F'. The Hex Dump pane shows the corresponding data bytes: 73 65 6E 73 sens, 6F 72 20 68 or h, 61 73 20 64 as d, 61 74 61 20 ata., 72 65 61 64 read, 79 3A 20 30 y: 0, 78 37 41 35 x7A5, 46 20 0D 0A F...

No.	Time	Du...	Summary	NWK Src	NWK ...	P#	M...	E#	Status
30	18.639...	0...	Association	AE12	0000	6			
51	58.408...	0...	Route Discovery	0000	FFFF	10			
61	58.921...	0...	Multicast	0000	FFFF	3			
73	60.080...	0...	Transport Datagram	EE07	0000	4			
90	62.600...	0...	Transport Datagram	AE12	0000	4			
111	79.711...	0...	Transport Datagram	EE07	0000	4			

No.	Time	Type	Summary	MAC Src	MAC Dest	Status
108	74.64...	APITr...	emberPermitJoining	ember00		
109	79.69...	Printf	sensor has data ready: 0x7A5F	ember01		
110	79.70...	APITr...	emberSendDatagram	ember01		
114	79.71...	Packet	Transport Datagram	EE07	0000	
115	79.71...	Packet	802.15.4 Ack	0000	EE07	
116	79.71...	APITr...	emberIncomingMessageHandler	ember00		

Sensor node transmits data to the sink

Events 110 through 122 encompass the transmission and receipt of a sensor node's message. Events 110 through 117 comprise the events that relate to transmission:

Event 110

Shows an API Trace to `emberSendDatagram()`, where sensor node `ember01` generates data `0x7A5F` and sends this message to its sink node.

Event 114

Shows the over-the-air traffic that results from the call to `emberSendDatagram()`. The Event Detail pane shows that the message is broken up into 802.15.4 and ZigBee layers.

Two additional layers are shown:

- The Ember Transport layer shows the various message parameters that are specific to Ember's transport layer.
- The Sensor-Sink Decoder offers a preview a future enhancement, where InSight Desktop users can write custom decoders that are specific to their applications. In this case, it contains two pieces of data: the sender node's EU164 and the sensor data itself.

The screenshot displays the InSight Desktop interface for a 'Sensor-sink demo'. The top pane shows a map of nodes with a transaction (No. 114) highlighted in purple, indicating a 'Transport Datagram' sent from node 0xAE12 to node 0x0000. The 'Event Detail' pane on the right provides a hierarchical view of the message structure, including the IEEE 802.15.4 layer, ZigBee Network, ZigBee Application Support, ZigBee Application Framework, and Ember Transport. The Ember Transport section shows a Message Identifier of 0xAD, Frame Control of 0x00, Message Type of Datagram (0), and a Destination EU164 of 000D6F000005B037. The 'Hex Dump' pane at the bottom right shows the raw data of the message, including the destination address and the application payload.

Event 115

Represents the 802.15.4 layer ACK. This tells the sender that the message was received one hop away. The Map pane depicts this event as a thin green arrow within the same transaction (thick purple) arrow.

Event 116

The stack notifies the application that a packet has arrived. With this API trace, you can:

- Verify that the stack is passing messages intended for this node into the application.
- Estimate how much time the stack requires to process the message. Processing time depends on how often the application calls `emberTick()`. In this case, the application starts processing the message data approximately 1.6 mS after receiving the message over the air.

Event 117

Traces the data from the sensor to the sink application via a `printf` statement from the sink application.

Non-contiguous transaction events

Events 114 and 122 mark the beginning and end points of a single Transport Datagram transaction. Not all the events between events 114 and 122 belong to this transaction, as indicated by breaks in the bracket line. The intervening events—116, 117, and 118—occurred during the transaction but are not considered part of it.

Receipt of sensor data is acknowledged

Although the session file shows that the receiving node received the data, the source node so far remains unaware of this. In multi-hop wireless networks, it is not unusual for individual packets to be dropped en route. For this reason, the source node awaits confirmation of message receipt from Ember's transport layer before it stops transmitting the message data, as shown in events 118 through 123:

Event 118

The Ember transport layer calls the datagram reply API in order to verify to the source node that the application received and handled the message data. This tells the transport mechanism on the source node that it can stop sending this particular message. This verification is especially important when the source and destination nodes are separated by multiple hops; in this case, a higher layer end-to-end ACK is required in order to verify that the data was transmitted.

Events 121 and 122

The transport layer acknowledgement (ACK) is transmitted to the source node.

The screenshot displays the InSight Desktop interface for a 'Sensor-sink demo'. The top section shows a network map with two nodes: 0xAE12 (PAN: 0x01FF, 000D6F000005AE69) and 0x0000 (PAN: 0x01FF, 000D6F000005B030). A red arrow indicates a packet being sent from 0xAE12 to 0x0000. Below the map is a 'Transactions' table with 28 entries shown. The 'Events' table below that shows 460 entries, with event 123 highlighted. The 'Event Detail' pane on the right shows the structure of an IEEE 802.15.4 frame, including ZigBee Network, Frame Control, Frame Type, Protocol Version, Discover Route, Security, Destination Address, Source Address, Radius, Sequence, ZigBee Application Support, ZigBee Application Framework, Ember Transport, and Message Identifier. The 'Hex Dump' pane at the bottom right shows the raw hex data of the frame.

No.	Time	Du...	Summary	NWK Src	NWK ...	P#	M...	E#	Status
111	79.711...	0...	Transport Datagram	EE07	0000	4			
126	82.246...	0...	Transport Datagram	AE12	0000	4			
144	99.360...	0...	Transport Datagram	EE07	0000	4			
162	101.86...	0...	Transport Datagram	AE12	0000	4			
207	118.97...	0...	Transport Datagram	EE07	0000	4			
243	138.60...	0...	Transport Datagram	EE07	0000	4			

No.	Time	Type	Summary	MAC Src	MAC Dest	Status
118	79.74...	APITr...	emberSendReply	ember00		
121	79.75...	Packet	Transport Datagram Reply	0000	EE07	
122	79.75...	Packet	802.15.4 Ack	EE07	0000	
123	79.75...	APITr...	emberMessageSent	ember01		
124	82.21...	Printf	sensor has data ready: 0x4...	ember02		
125	82.23...	APITr...	emberSendDatagram	ember02		
129	82.24...	Packet	Transport Datagram	AE12	0000	
130	82.24...	Packet	802.15.4 Ack	0000	AE12	

```

IEEE 802.15.4
ZigBee Network
  Frame Control: 0x0004
  Frame Type: Data (0)
  Protocol Version: 0x01
  Discover Route: Suppress (0)
  Security: false
  Destination Address: 0xEE07
  Source Address: 0x0000
  Radius: 0x0A
  Sequence: 0x10
ZigBee Application Support
ZigBee Application Framework
Ember Transport
  Message Identifier: 0xAD
Hex Dump
1C 61 88 7E .a.~
FF 01 07 EE . . . .
00 00 04 00 . . . .
07 EE 00 00 . . . .
0A 10 00 01 . . . .
00 0F C0 01 . . . .
E1 AD 04 . . . .

```

Event 123

Call to API trace `emberMessageSent()`. This API confirms that the message was received by the intended destination node, relieving the application from having to generate its own mechanism to reliably transport data from one node to another across the network.

Perfect trace capture versus sniffer capture

The sensor-sink demo session represents a perfect trace capture, where all messages that are processed by network nodes are relayed to InSight Desktop via their InSight Ports. Each EM250 and EM260 node has a built-in InSight Port, which is integrated with a dedicated packet trace circuit that intercepts all packet information. This non-invasive technology enables InSight Desktop to capture all node data, including failed transmissions, and debug messages.

Capturing from all nodes in a network yields a perfect trace session: the session data compiles all incoming and outgoing data from each node in chronological real time, providing a richly layered view of all activity within a network. This can be especially useful for debugging a network while it undergoes development.

A sniffer capture intercepts all over-the-air transmissions among network nodes; however, the sniffer cannot detect any data transmissions that are known only to their senders, such as messages that fail to arrive at their destination and debug data such as API traces. Sniffers are especially fallible in large networks where transmissions often require multiple hops.

Using filters

Even simple transactions can comprise a large number of events, and often only a few of these are of interest. In order to facilitate analysis, InSight Desktop provides a number of built-in filters that are accessible from menus. You can also compose your own filter expressions for a given session.

Menu-accessible filters

InSight Desktop provides two types of filters that are accessible as menu options:

- Quick filters that are accessible by selecting an event type.
- Saved filters that are accessible from the Filters menu.

Quick filters

You can specify to exclude events of a given type as follows:

1. Select any event of that type.

For example, select a `printf` event.

2. Right click on the event,
3. From the context menu choose the **Hide type** option.

For example, if a `printf` event is selected, you can choose the context menu option **Hide type: Printf**.

When you set a quick filter, InSight Desktop updates the filter bar with the corresponding filter expression. For example, if you choose the quick filter **Hide type: Printf**, the filter bar displays this expression:

```
!isType(Printf)
```

Saved filters

InSight Desktop maintains a set of saved filters that you can create or modify in the Filter Manager. Through the Filter Manager, you can specify which saved filters to display on the Filters menu.

As installed, InSight Desktop displays one saved filter from the Filters menu, **Hide Neighbor Exchange**. You turn this filter on and off by toggling the menu option. When the filter is on, InSight Desktop hides all neighbor exchange events from the Events pane.

Filter bar expressions

A session's filter bar lets you build filter expressions in two ways:

- Enter the expression directly in the filter bar field, and press Return.
- Use the Expression Builder by clicking Compose

Either way, you can compose complex logical statements for filtering events. The following steps show how to use the Expression Builder to build an expression that shows only messages sent to node 0x0000:

1. Select any Packet type event—for example, event 147.
2. Right-click on the event, and from the context menu choose **Show only type: Packet**

The filter bar now shows this expression: `isType(Packet)`

3. Modify this expression by clicking on the Filter Bar's Compose button. This invokes the Expression Builder dialog.
4. At the end of the expression, type the AND operator `&&`
5. In the Field pane, expand 802.15.4 and choose Short Destination Address
6. After the `==` operator, type `0x0000`

The complete expression should look like this:

```
isType(Packet) && fifteenFour.dest == 0x0000
```

7. Click **OK**

The filter allows the Events pane to display only Packet-type events where the message destination is node 0x0000:

Transactions total:28 shown:22										
No.	Time	Du...	Summary	NWK Src	NWK ...	P#	M...	E#	Status	
14	11.647...	0...	Association	EE07	0000	6				
30	18.639...	0...	Association	AE12	0000	6				
73	60.080...	0...	Transport Datagram	EE07	0000	4				
90	62.600...	0...	Transport Datagram	AE12	0000	4				
111	79.711...	0...	Transport Datagram	EE07	0000	4				
126	82.246...	0...	Transport Datagram	AE12	0000	4				
144	99.360...	0...	Transport Datagram	EE07	0000	4				

Events total:460 shown:24							
No.	Time	Type	Summary	MAC Src	MAC Dest	Status	
16	11.64...	Packet	Association Request	000D6F...	0000		
19	11.68...	Packet	Data Request	000D6F...	0000		
32	18.63...	Packet	Association Request	000D6F...	0000		
35	18.67...	Packet	Data Request	000D6F...	0000		
76	60.08...	Packet	Transport Datagram	EE07	0000		
93	62.60...	Packet	Transport Datagram	AE12	0000		

Copyright © 2006 by Ember Corporation

All rights reserved.

The information in this document is subject to change without notice. The statements, configurations, technical data, and recommendations in this document are believed to be accurate and reliable but are presented without express or implied warranty. Users must take full responsibility for their applications of any products specified in this document. The information in this document is the property of Ember Corporation.

Title, ownership, and all rights in copyrights, patents, trademarks, trade secrets and other intellectual property rights in the Ember Proprietary Products and any copy, portion, or modification thereof, shall not transfer to Purchaser or its customers and shall remain in Ember and its licensors.

No source code rights are granted to Purchaser or its customers with respect to all Ember Application Software. Purchaser agrees not to copy, modify, alter, translate, decompile, disassemble, or reverse engineer the Ember Hardware (including without limitation any embedded software) or attempt to disable any security devices or codes incorporated in the Ember Hardware. Purchaser shall not alter, remove, or obscure any printed or displayed legal notices contained on or in the Ember Hardware.

Ember, Ember Enabled, EmberZNet, InSight, and the Ember logo are trademarks of Ember Corporation.

All other trademarks are the property of their respective holders.

ember

Ember Corporation
343 Congress Street
Boston MA 02210
617.951.0200
www.ember.com



wireless semiconductor solutions