



# **Cortina Systems® IXF1010/IXF1110 Media Access Controller Application Programmer Interface**

**User Guide**

---

**30 January 2007**

**Document Number 280053**

**Revision 2.0**

---

This document contains information proprietary to Cortina Systems, Inc. Any use or disclosure, in whole or in part, of this information to any unauthorized party, for any purposes other than that for which it is provided is expressly prohibited except as authorized by Cortina Systems, Inc. in writing. Cortina Systems, Inc. reserves its rights to pursue both civil and criminal penalties for copying or disclosure of this material without authorization.

\*Other names and brands may be claimed as the property of others.

© Cortina Systems, Inc. 2007

---

## Contents

<b>1.0 Document Overview .....</b>	<b>3</b>
<b>2.0 Build Options .....</b>	<b>5</b>
<b>3.0 API Definitions .....</b>	<b>6</b>
3.1 API Function Description Format .....	6
3.2 IXF1110 API—Function Prototypes .....	6

## API Functions

Ixf1110GetChipInfo .....	7
Ixf1110GetCounters .....	8
Ixf1110Reset .....	9
Ixf1110SetCfg .....	10
Ixf1110GetCfg .....	11
Ixf1110GetStatus .....	12
Ixf1110GetBuildVersion .....	13
Ixf1110Read .....	14
Ixf1110Write .....	15
Ixf1110GetMacAddress .....	16
Ixf1110SetMacAddress .....	17
Ixf1110SetAlarmCfg .....	18
Ixf1110InitAlarmCallback .....	19
Ixf1110Chiplsr .....	20
Ixf1110InitAllocMemory .....	21
Ixf1110DeAllocMemory .....	22
Ixf1110InitChip .....	23
IxfApiSetChipOffline .....	24
IxfApiSetChipOnline .....	25

## Tables

1 IXF1010 MAC API Functions and Syntax .....	3
2 Compiler Build Options .....	5
3 API Description Format .....	6

## Revision History

<b>Revision 2.0</b> <b>Revision Date: 30 January 2007</b>
First Release from Cortina Systems, Inc.

<b>Revision 001</b> <b>Revision Date: 01 March 2004</b>
Initial release

## 1.0 Document Overview

This document describes the Cortina Systems® IXF1010/IXF1110 Media Access Controller Application Programmer Interface (API), which provides initialization and access to the IXF1010/IXF1110 MAC devices. In addition, this document details the build options available when compiling the driver.

**Note:** Section 3.2, *IXF1110 API—Function Prototypes*, on page 6 describes the IXF1110 MAC API functions. The IXF1010 MAC has the same function calls as the IXF1110 MAC except the function name and syntax contains *lxf1010* instead of *lxf1110*. Table 1 contains a list of IXF1010 MAC functions and syntax.

**Table 1 IXF1010 MAC API Functions and Syntax (Sheet 1 of 2)**

IXF1010 MAC		IXF1110 MAC Functions
Function Name	Syntax	
lxf1010GetChipInfo	extern bb_Error_e lxf1010GetChipInfo ( bb_ChipData_t *pChipData, bb_ChipInfo_t *pChipInfo );	See <a href="#">lxf1110GetChipInfo</a> , on page 7
lxf1010GetCounters	extern bb_Error_e lxf1010GetCounters ( bb_ChipData_t *pChipData, bb_ChipSegment_t *ptSegment, bb_SelCounters_e eCounter, void *pCounters );	See <a href="#">lxf1110GetCounters</a> , on page 8
lxf1010Reset	extern bb_Error_e lxf1010Reset ( bb_ChipData_t *pChipData, bb_ChipSegment_t *ptSegment, bb_SelResetType_e resetType );	See <a href="#">lxf1110Reset</a> , on page 9
lxf1010SetCfg	extern bb_Error_e lxf1010SetCfg ( bb_ChipData_t *pChipData, bb_ChipSegment_t *ptSegment, bb_SelConfig_e SelCfg );	See <a href="#">lxf1110SetCfg</a> , on page 10
lxf1010GetCfg	extern bb_Error_e lxf1010GetCfg ( bb_ChipData_t *pChipData, bb_ChipSegment_t *ptSegment, bb_SelConfig_e SelCfg );	See <a href="#">lxf1110GetCfg</a> , on page 11
lxf1010GetStatus	extern bb_Error_e lxf1010GetStatus ( bb_ChipData_t *pChipData, bb_ChipSegment_t *ptSegment, bb_SelStatus_e selStatus, void *pStatus );	See <a href="#">lxf1110GetStatus</a> , on page 12
lxf1010GetBuildVersion	extern bb_Error_e lxf1010GetBuildVersion ( bb_ChipData_t *pChipData, char *drvName, char *date, ushort *buildVer, ushort *buildRev );	See <a href="#">lxf1110GetBuildVersion</a> , on page 13
lxf1010Read	extern bb_Error_e lxf1010Read ( bb_ChipData_t *pChipData, bb_Word_Size_t wordSize, ulong address, ushort length, void *buffer );	See <a href="#">lxf1110Read</a> , on page 14
lxf1010Write	extern bb_Error_e lxf1010Write ( bb_ChipData_t *pChipData, bb_Word_Size_t wordSize, ulong address, ushort length, void *buffer );	See <a href="#">lxf1110Write</a> , on page 15
lxf1010GetMacAddress	extern bb_Error_e lxf1010GetMacAddress ( bb_ChipData_t *pChipData, bb_ChipSegment_t *section, void *address );	See <a href="#">lxf1110GetMacAddress</a> , on page 16
lxf1010SetMacAddress	extern bb_Error_e lxf1010SetMacAddress ( bb_ChipData_t *pChipData, bb_ChipSegment_t *section, void *address );	See <a href="#">lxf1110SetMacAddress</a> , on page 17
lxf1010SetAlarmCfg	extern bb_Error_e lxf1010SetAlarmCfg ( bb_ChipData_t *pChipData, bb_ChipSegment_t *section, bb_AlarmType_e AlarmType, void *pAlarmCfg );	See <a href="#">lxf1110SetAlarmCfg</a> , on page 18
lxf1010InitAlarmCallback	extern bb_Error_e lxf1010InitAlarmCallback ( bb_ChipData_t *pChipData, AlarmCallBack pAlarmCallbackAr );	See <a href="#">lxf1110InitAlarmCallback</a> , on page 19
lxf1010Chiplsr	extern bb_Error_e lxf1010Chiplsr ( bb_ChipData_t *pChipData );	See <a href="#">lxf1110Chiplsr</a> , on page 20
lxf1010InitAllocMemory	extern bb_Error_e lxf1010InitAllocMemory ( bb_ChipData_t *pChipData );	See <a href="#">lxf1110InitAllocMemory</a> , on page 21

**Table 1 IXF1010 MAC API Functions and Syntax (Sheet 2 of 2)**

IXF1010 MAC		IXF1110 MAC Functions
Function Name	Syntax	
Ixf1010DeAllocMemory	extern bb_Error_e Ixf1010DeAllocMemory ( bb_ChipData_t *pChipData );	See <i>Ixf1110DeAllocMemory</i> , on page 22
Ixf1010InitChip	extern bb_Error_e Ixf1010InitChip ( bb_ChipData_t *pChipData, InitRegTable_t *pTable );	See <i>Ixf1110InitChip</i> , on page 23
IxfApiSetChipOffline	bb_Error_e Ixf1010SetChipOffline ( bb_ChipData_t *pChipData, bb_ChipSegment_t *section );	See <i>IxfApiSetChipOffline</i> , on page 24
IxfApiSetChipOnline	bb_Error_e Ixf1010SetChipOnline ( bb_ChipData_t *pChipData, bb_ChipSegment_t *section );	See <i>IxfApiSetChipOnline</i> , on page 25

## 2.0 Build Options

Table 2 lists compile-time options associated with the IXF1010/IXF1110 MAC driver.

**Table 2** Compiler Build Options

Defined Symbol	Description
VXWORKS	Builds the driver to support VxWorks® OS
LITTLE_ENDIAN	Builds the driver for little endian processors
BIG_ENDIAN	Builds the driver for big endian processors
LITTLE_ENDIAN_BITFIELD	Builds the driver for compilers that implement bit-fields in a little endian manner
BIG_ENDIAN_BITFIELD	Builds the driver for compilers that implement bit-fields in a big endian manner
ETHERNET_INCLUDED	Builds the driver for the Ethernet feature set
INCLUDE_1110_LIB	Builds the IXF1110 library files
IXF_REG_POINTER_32	Typedef register pointer to an unsigned long
IXF_REG_DATA_32	Typedef register data type to be unsigned long
__KERNEL__	If building on linux platform
IXF_LINUX	If building on linux platform

## 3.0 API Definitions

### 3.1 API Function Description Format

This API section details each of the routines in the Common API Layer. [Table 3](#) shows the format of the API description.

**Table 3** API Description Format

API Function Name: Syntax: Purpose: Inputs: Outputs: Returns: Globals: Notes: See Also:
---

### 3.2 IXF1110 API—Function Prototypes

This section contains the common IXF API functions supported by the IXF1110 API. The IXF1010 MAC has the same function calls as the IXF1110 MAC except the function names contains *lxf1010* instead of *lxf1110*. The functions are:

- [lxf1110GetChipInfo, on page 7](#)
- [lxf1110GetCounters, on page 8](#)
- [lxf1110Reset, on page 9](#)
- [lxf1110SetCfg, on page 10](#)
- [lxf1110GetCfg, on page 11](#)
- [lxf1110GetStatus, on page 12](#)
- [lxf1110GetBuildVersion, on page 13](#)
- [lxf1110Read, on page 14](#)
- [lxf1110Write, on page 15](#)
- [lxf1110GetMacAddress, on page 16](#)
- [lxf1110SetMacAddress, on page 17](#)
- [lxf1110SetAlarmCfg, on page 18](#)
- [lxf1110InitAlarmCallback, on page 19](#)
- [lxf1110Chiplsr, on page 20](#)
- [lxf1110InitAllocMemory, on page 21](#)
- [lxf1110DeAllocMemory, on page 22](#)
- [lxf1110InitChip, on page 23](#)
- [lxfApiSetChipOffline, on page 24](#)
- [lxfApiSetChipOnline, on page 25](#)



---

## Ixf1110GetCounters

**Syntax:** extern bb\_Error\_e **Ixf1110GetCounters** ( bb\_ChipData\_t \*pChipData,  
bb\_ChipSegment\_t \*ptSegment,  
bb\_SelCounters\_e eCounter,  
void \*pCounters );

**Purpose:** The **IXF1110GetCounters** function retrieves one/all counters (from one/all) blocks of the IXF1110 device.

**Inputs:**

<b>pChipData</b>	Initialized chip data
<b>section</b>	Determines the section (block) to retrieve counters
<b>eCounter</b>	Specifies which counter
<b>pCounters</b>	Buffer to write the counters

**Outputs:** None.

**Returns:** **bb\_NO\_ERROR**  
**bb\_INV\_CHIP\_SEGMENT**  
**bb\_INV\_BLOCK\_OPERATION**  
**bb\_NULL\_ARG**

**Globals:** None.

**Notes:** **pCounters** will either point to a structure to receive gathered counters or a **bb\_RegDataValue\_type** to receive value of a single counter.

**See Also:** Refer to structure: **bb\_SelCounters\_e** in file **ixf\_api\_d.h** for specific counters.

---

## Ixf1110Reset

**Syntax:** extern bb\_Error\_e Ixf1110Reset ( bb\_ChipData\_t \*pChipData,  
bb\_ChipSegment\_t \*ptSegment,  
bb\_SelResetType\_e resetType );

**Purpose:** The Ixf1110Reset function resets the chip or a section of the chip, then reconfigures it.

**Inputs:**

pChipData	Initialized chip data
section	Determines the section (block) to reset
resetType	Specifies type of Reset

**Outputs:** None.

**Returns:** bb\_NO\_ERROR  
bb\_INV\_RESET\_TYPE  
bb\_INV\_CHIP\_TYPE

**Globals:** None.

**Notes:** resetType can be:  
bb\_RESET\_RX\_FIFO  
bb\_RESET\_TX\_FIFO  
bb\_RESET\_XGMAC  
bb\_RESET\_CORE\_CLK  
bb\_RESET\_XGMAC\_ALL  
bb\_RESET\_RX\_FIFO\_ALL  
bb\_RESET\_TX\_FIFO\_ALL  
If you choose *not* to reset *all*, you must supply port number in the ptSegment.group.channel.

**See Also:** None.

---

## Ixf1110SetCfg

**Syntax:** extern bb\_Error\_e Ixf1110SetCfg ( bb\_ChipData\_t \*pChipData,  
bb\_ChipSegment\_t \*ptSegment,  
bb\_SelConfig\_e SelCfg );

**Purpose:** The Ixf1110SetCfg function configures one/all block(s) of IXF1110 device.

**Inputs:**

*pChipData	Pointer to the chip information
*pChipSeg	Pointer to chip segment structure
SelCfg	The enumeration value that defines what configuration needs to be set

**Outputs:** None.

**Returns:** bb\_NO\_ERROR  
bb\_NULL\_ARG  
bb\_INV\_PARAMETER  
bb\_INV\_CHIP\_TYPE

**Globals:** None.

**Notes:** ptSegment->block.block can be set to any of the following:

ixf\_eGLOBAL  
ixf\_eSPI4  
ixf\_eXGMAC  
ixf\_eGLBRX  
ixf\_eGLBTX  
ixf\_eSERDES  
ixf\_eGBIC

selCfg must be set to e\_ALL\_CONFIG

If you choose the ixf\_eXGMAC function, you must supply the port number in the ptSegment.group.channel.

**See Also:** Ixf1110SetCfg

---

## Ixf1110GetCfg

**Syntax:** extern bb\_Error\_e Ixf1110GetCfg ( bb\_ChipData\_t \*pChipData,  
bb\_ChipSegment\_t \*ptSegment,  
bb\_SelConfig\_e SelCfg );

**Purpose:** The Ixf1110GetCfg function reads the configuration of one/all block(s) of IXF1110 device.

**Inputs:**

*pChipData	Pointer to the chip information
*pChipSeg	Pointer to chip segment structure
SelCfg	The enumeration value that defines what configuration needs to be retrieved

**Outputs:** Chip configuration structure within pChipData will be filled with appropriate configuration retrieved from device

**Returns:** bb\_NO\_ERROR  
bb\_NULL\_ARG  
bb\_INV\_PARAMETER  
bb\_INV\_CHIP\_TYPE

**Globals:** None.

**Notes:** ptSegment->block.block can be set to any of the following:

ixf\_eGLOBAL  
ixf\_eSPI4  
ixf\_eXGMAC  
ixf\_eGLBRX  
ixf\_eGLBTX  
ixf\_eSERDES  
ixf\_eGBIC  
selCfg must be set to e\_ALL\_CONFIG

**See Also:** [Ixf1110SetCfg](#)

---

## Ixf1110GetStatus

**Syntax:** extern bb\_Error\_e Ixf1110GetStatus ( bb\_ChipData\_t \*pChipData,  
bb\_ChipSegment\_t \*ptSegment,  
bb\_SelStatus\_e selStatus,  
void \*pStatus );

**Purpose:** The Ixf1110GetStatus function reads the status of one/all block(s) of IXF1110 device.

**Inputs:**

pChipData	Initialized chip data
section	Determines the section (block) to get the status from
selStatus	Type of Status

**Outputs:** pStatus Place to put Status

**Returns:** bb\_Error\_e

**Globals:** None.

**Notes:** ptSegment->block.block can be set to any of the following:

ixf\_eSPI4  
ixf\_eXGMAC  
ixf\_eGLBRX  
ixf\_eGLBTX  
ixf\_eGBIC

selStatus must be set to ixf\_eALL\_STATUS

If you choose the ixf\_eXGMAC function, you must supply the port number in:  
ptSegment.group.channel.

**See Also:** None.

---

## Ixf1110GetBuildVersion

**Syntax:** extern bb\_Error\_e Ixf1110GetBuildVersion ( bb\_ChipData\_t \*pChipData,  
char \*drvName,  
char \*date,  
ushort \*buildVer,  
ushort \*buildRev );

**Purpose:** The Ixf1110GetBuildVersion function reads the build version for the IXF1110 device driver.

**Inputs:** bb\_ChipData\_t \*pChipData  
char \*drvName  
char \*date  
ushort \*buildVer  
ushort \*buildRev

**Outputs:** buildVer  
buildRev  
date  
drvName

**Returns:** bb\_NO\_ERROR

**Globals:** None.

**Notes:** None.

**See Also:** None.

---

## Ixf1110Read

**Syntax:** extern bb\_Error\_e **Ixf1110Read** ( bb\_ChipData\_t \*pChipData,  
bb\_Word\_Size\_t wordSize,  
ulong address,  
ushort length,  
void \*buffer );

**Purpose:** The **Ixf1110Read** function reads *n* number of one/two/four byte values to the device.

**Inputs:**

<b>pChipData</b>	Initialized chip data
<b>wordSize</b>	Four-byte word size
<b>address</b>	Starting address to read from
<b>length</b>	Number of words to read
<b>buffer</b>	Location where words are read into

**Outputs:** None.

**Returns:** **bb\_Error\_e**  
**bb\_WORD\_SIZE\_UNSUPPORTED**

**Globals:** None.

**Notes:** Input variable **wordSize** must be set to **FOUR\_BYTES**.

**See Also:** [Ixf1110Write](#)

---

## Ixf1110Write

**Syntax:** extern bb\_Error\_e **Ixf1110Write** ( bb\_ChipData\_t \*pChipData,  
bb\_Word\_Size\_t wordSize,  
ulong address,  
ushort length,  
void \*buffer );

**Purpose:** The **Ixf1110Write** function writes *n* number of one/two/four byte values to the device.

**Inputs:**

<b>pChipData</b>	Initialized chip data
<b>wordSize</b>	Four-byte word size
<b>address</b>	Starting address to write to
<b>length</b>	Number of words to write
<b>buffer</b>	Location where words are written from

**Outputs:** None.

**Returns:** **bb\_Error\_e**  
**bb\_WORD\_SIZE\_UNSUPPORTED**

**Globals:** **bb\_WORD\_SIZE\_UNSUPPORTED**

**Notes:** Input variable **wordSize** must be set to **FOUR\_BYTES**.

**See Also:** [Ixf1110Read](#)

---

## Ixf1110GetMacAddress

**Syntax:** extern bb\_Error\_e **Ixf1110GetMacAddress** ( bb\_ChipData\_t \*pChipData,  
bb\_ChipSegment\_t \*section,  
void \*address );

**Purpose:** The **Ixf1110GetMacAddress** function gets the MAC Address of the device.

**Inputs:** **pChipData** Initialized chip data  
**section** Determines the section (block) of the chip

**Outputs:** **address** Pointer to the MAC Address structure

**Returns:** **bb\_Error\_e** Returns error condition if not successful

**Globals:** None.

**Notes:** None.

**See Also:** [Ixf1110SetMacAddress](#)

---

## Ixf1110SetMacAddress

**Syntax:**

```
extern bb_Error_e Ixf1110SetMacAddress ( bb_ChipData_t *pChipData,  
                                         bb_ChipSegment_t *section,  
                                         void *address );
```

**Purpose:** The `Ixf1110SetMacAddress` function sets the MAC Address of the device.

**Inputs:**

<code>pChipData</code>	Initialized chip data
<code>section</code>	Determines the section (block) of the chip
<code>address</code>	Pointer to the MAC Address structure

**Outputs:** `address` Pointer to the MAC Address structure

**Returns:** None

**Globals:** None.

**Notes:** None.

**See Also:** [Ixf1110GetMacAddress](#)

---

## Ixf1110SetAlarmCfg

- Syntax:**

```
extern bb_Error_e Ixf1110SetAlarmCfg ( bb_ChipData_t *pChipData,
                                       bb_ChipSegment_t *section,
                                       bb_AlarmType_e AlarmType,
                                       void *pAlarmCfg );
```
- Purpose:** The `Ixf1110SetAlarmCfg` function configures the interrupt masks.
- Inputs:**
- |                                       |  |
|---------------------------------------|--|
| <code>pChipData</code>                | Initialized chip data                      |
| <code>section</code>                  | Determines the section (block) of the chip |
| <code>address</code>                  | Pointer to the MAC Address structure       |
| <code>bb_AlarmType_e AlarmType</code> | Type of alarm to configure                 |
| <code>void *pAlarmCfg</code>          | Not used                                   |
- Outputs:** None.
- Returns:** `bb_Error_e` Returns error condition if not successful
- Globals:** None.
- Notes:** AlarmType can be either:  
`bb_1110_RX_LOS_ALARM`  
`bb_1110_TX_FAULT_ALARM`  
`bb_1110_MOD_DEF_ALARM`
- See Also:** For further information on the `RX_LOS`, `TX_FAULT` and `MOD_DEF` interrupts, refer to the IXF1110 datasheet.

---

## Ixf1110InitAlarmCallback

**Syntax:** `extern bb_Error_e Ixf1110InitAlarmCallback ( bb_ChipData_t *pChipData,  
AlarmCallBack pAlarmCallbackAr );`

**Purpose:** The `Ixf1110InitAlarmCallback` function initialize the alarm callback function with the device driver.

**Inputs:**

<code>pChipData</code>	Validated chip data
<code>pAlarmCallback</code>	Alarm Callback

**Outputs:** None.

**Returns:** `bb_NO_ERROR`

**Globals:** None.

**Notes:** None.

**See Also:** None.

---

## Ixf1110ChipIsr

**Syntax:** `extern bb_Error_e Ixf1110ChipIsr ( bb_ChipData_t *pChipData );`

**Purpose:** The `Ixf1110ChipIsr` (ISR) function services the GBIC interrupts that the user selects when monitoring the [Ixf1110SetAlarmCfg](#).

**Inputs:** `pChipData` Validated chip data

**Outputs:** None.

**Returns:** `bb_INV_CHIP_TYPE`  
`bb_NO_ERROR`

**Globals:** None.

**Notes:** None.

**See Also:** None.

---

## Ixf1110InitAllocMemory

**Syntax:** `extern bb_Error_e Ixf1110InitAllocMemory ( bb_ChipData_t *pChipData );`

**Purpose:** The `Ixf1110InitAllocMemory` function allocates the memory space for device driver configuration and alarm structures.

**Inputs:** `pChipData` Validated chip data

**Outputs:** None.

**Returns:** `bb_NO_ERROR`

**Globals:** None.

**Notes:** None.

**See Also:** [Ixf1110DeAllocMemory](#)

---

## Ixf1110DeAllocMemory

**Syntax:** `extern bb_Error_e Ixf1110DeAllocMemory ( bb_ChipData_t *pChipData );`

**Purpose:** The `Ixf1110DeAllocMemory` function de-allocates the memory space for device driver configuration and alarm structures.

**Inputs:** `pChipData` Validated chip data

**Outputs:** None.

**Returns:** `bb_NO_ERROR`  
`bb_NO_CHIP_CFG`

**Globals:** None.

**Notes:** None.

**See Also:** [Ixf1110InitAllocMemory](#)



---

## IxfApiSetChipOffline

**Syntax:** `bb_Error_e Ixf1110SetChipOffline ( bb_ChipData_t *pChipData,  
bb_ChipSegment_t *section );`

**Purpose:** The `Ixf1110SetChipOffline` function disables the port selected in the section parameter: `section.group.channel` = port to disable. This is achieved by clearing the appropriate bit in the Port Enable Register.

**Inputs:**

<code>pChipData</code>	Validated chip data
<code>section</code>	Chip segment

**Outputs:** None.

**Returns:** `bb_NO_ERROR`  
`bb_NULL_ARG`

**Globals:** None.

**Notes:** None.

**See Also:** [IxfApiSetChipOnline](#)

---

## IxfApiSetChipOnline

**Syntax:** `bb_Error_e Ixf1110SetChipOnline ( bb_ChipData_t *pChipData,  
bb_ChipSegment_t *section );`

**Purpose:** The `Ixf1110SetChipOnline` function enables the port selected in the section parameter: `section.group.channel` = port to disable. This is achieved by setting the appropriate bit in the Port Enable Register.

**Inputs:**

<code>pChipData</code>	Validated chip data
<code>section</code>	Chip segment

**Outputs:** None.

**Returns:** `bb_NO_ERROR`  
`bb_NULL_ARG`

**Globals:** None.

**Notes:** None.

**See Also:** [IxfApiSetChipOffline](#)

---

## Contact Information

**Cortina Systems, Inc.**  
**840 W. California Ave**  
**Sunnyvale, CA 94086**

**408-481-2300**

**[sales@cortina-systems.com](mailto:sales@cortina-systems.com)**

**[apps@cortina-systems.com](mailto:apps@cortina-systems.com)**

**[www.cortina-systems.com](http://www.cortina-systems.com)**

---

This document contains information proprietary to Cortina Systems, Inc. Any use or disclosure, in whole or in part, of this information to any unauthorized party, for any purposes other than that for which it is provided is expressly prohibited except as authorized by Cortina Systems, Inc. in writing. Cortina Systems, Inc. reserves its rights to pursue both civil and criminal penalties for copying or disclosure of this material without authorization.

\*Other names and brands may be claimed as the property of others.

© Cortina Systems, Inc. 2007