



iChip™ CO2064

Ver. i2064/720B03

Release Notes

October 2007

Table of Contents

Table of Contents	2
What's New in This Version	3
Two Firmware Flavors, Four Different Versions	3
Loading the Firmware from Host	3
Loading the Firmware from SPI Flash Memory	4
SSL3/TLS1 Secure Socket Protocol Support in Hardware	6
Establishing an SSL3/TLS1 Socket Connection.....	7
Sending and Receiving Data over an SSL3/TLS1 Socket	7
SSL3/TLS1 Handshake and Session Example	7
Secure Socket Protocol Parameters	9
+iCS — Define the SSL3/TLS1 Cipher Suite	9
+iCA — Define SSL3/TLS1 Certificate Authority	10
+iCERT — Define SSL3/TLS1 Certificate	12
+iPKEY — Define iChip's Private Key	13
Known Bugs	15

What's New in This Version

This section describes all the new features included in the current firmware version.

Two Firmware Flavors, Four Different Versions

The current firmware version comes in two flavors: 'A' and 'B'.

Flavor 'A' supports the following features:

- 10 active TCP/UDP sockets and two listening sockets
- SerialNET mode
- Sending plain-text e-mail messages over the SMTP protocol
- An FTP client
- An HTTP client

Flavor 'B' supports the following features:

- A single secure SSL3/TLS1 socket
- 9 non-secure TCP/UDP sockets
- Network Time Client

Each of these flavors comes in two firmware upload versions, RAM and SPI, for a total of 4 different versions. The RAM version is suitable for iChips that load their firmware directly from the host memory. The SPI version is intended for iChips that load their firmware from an external flash memory over an SPI interface. The table below lists the four different versions.

<i>Firmware Version No.</i>	<i>Flavor</i>	<i>FW Loaded from</i>
i2064/720B03aSpi	'A'	Flash
i2064/720B03aRam	'A'	Host
i2064/720B03bSpiSSL	'B'	Flash
i2064/720B03bRamSSL	'B'	Host

Loading the Firmware from Host

To load the iChip firmware from the host memory via the RS232 serial interface, you need to obtain the RAM version of the firmware flavor you wish to install from Connect One. The RAM version is provided as two separate files: a *version.hdr* file and a *version.dat* file. The following procedure assumes that you have a host application running on your PC, in which you insert appropriate lines of code, as detailed below.

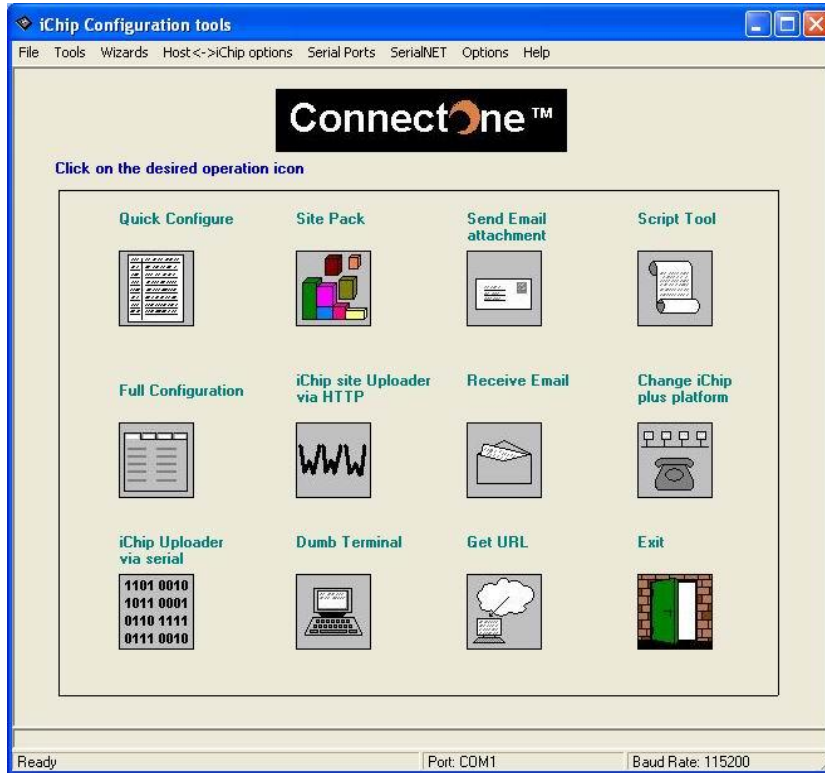
1. Power up the iChip and assert the MSEL pin for *at least* 5 seconds.
2. Use the connected UART at a baud rate between 9600 and 115200 to send a capital **U** character to the iChip. Wait for a **>** prompt.

3. Send a capital **L** and wait for a **>** prompt.
4. Send the *version.hdr* file and wait for a **>** prompt.
5. Send the *version.dat* file.

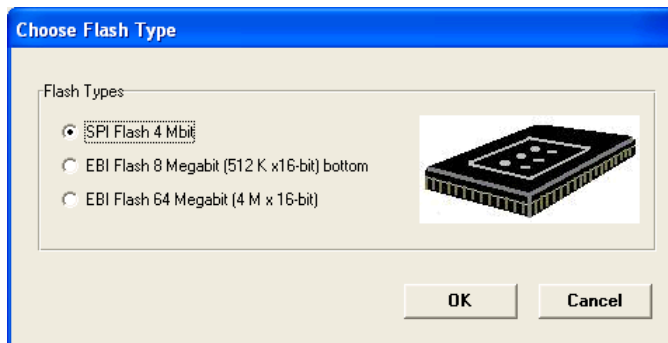
The firmware is now loaded into iChip's RAM and ready for use.

Loading the Firmware from SPI Flash Memory

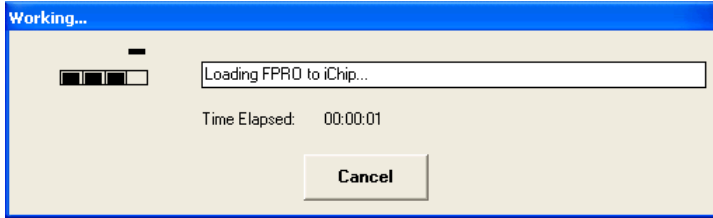
1. Invoke the iChipConfig utility.



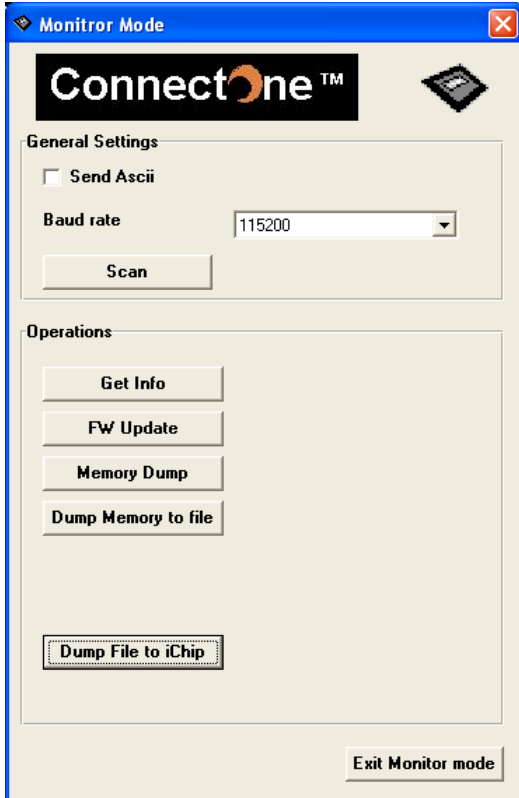
2. Click **Tools > CO2128/2064 Monitor Mode**.
3. In the dialog box displayed, select the *SPI Flash 4 Mbit* checkbox and click **OK**.



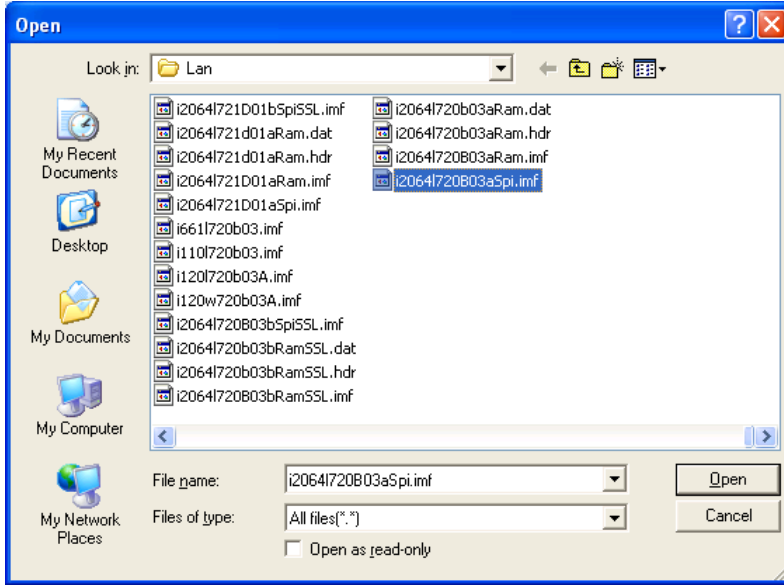
4. Wait for the utility to finish loading FPRO into iChip's RAM.



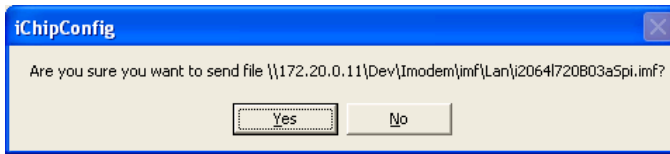
5. In the dialog box displayed, click the **FW Update** button.



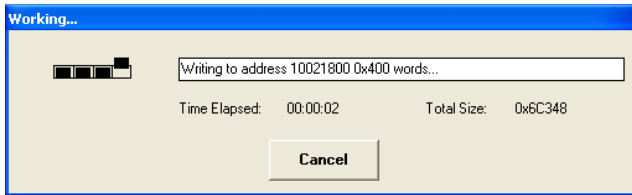
6. In the dialog box that appears, select the .imf image file of the firmware version you wish to upload and click **Open**.



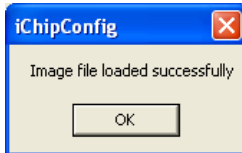
7. Click **Yes** when prompted.



8. The following screen appears:



9. **OK** the *Image file loaded successfully* dialog box.



The firmware is loaded into SPI flash memory. When you reset the iChip, it automatically loads the firmware from flash memory into its internal RAM.

SSL3/TLS1 Secure Socket Protocol Support in Hardware

The iChip 2064 supports the SSL3/TLS1 secure socket protocol in hardware, based on RFC2246. iChip supports the following Cipher suites:

- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA

Establishing an SSL3/TLS1 Socket Connection

iChip supports a single secure SSL3/TLS1 active socket connection. Opening an SSL3/TLS1 socket on iChip involves two steps:

1. Open a standard TCP/IP socket to an SSL3 server.
2. Initiate an SSL3 handshake over the open socket to establish an SSL3 session. SSL3 handshake negotiations are initiated using the AT+iSSL command.

iChip negotiates the secure connection based on several SSL3 related parameters. It authenticates the remote SSL3 server by verifying that the server's certificate is signed by a trusted Certificate Authority (CA). The trusted CA's certificate is stored in iChip's CA parameter. Following a successful SSL3 handshake, iChip encrypts all data sent across the socket according to the cipher suite and keys agreed upon during the SSL3 handshake. Data received on the socket is decrypted by iChip prior to making it available to the host processor.

Sending and Receiving Data over an SSL3/TLS1 Socket

The AT+iSSND command is used to send data over an SSL3/TLS1 socket, using the same syntax as for non-secure sockets:

```
AT+iSSND[%]:<hn>,<size>:<data>
```

However, the *size* parameter is interpreted as the size of the data packet to encrypt. It is limited to 2K. Receiving data on an SSL3/TLS1 socket is carried out using the AT+iSRCV command. iChip automatically decrypts data that arrives on the SSL3/TLS1 socket. The data transferred to the host is always decrypted data.

SSL3/TLS1 Handshake and Session Example

Take for example an SSL3/TLS1 server at `secure.sslserver.com` running a secure application on port 1503. Using iChip, the following sequence opens a secure SSL3/TLS1 socket to that application and exchanges data securely. For clarity, commands sent to iChip appear in **bold** and iChip replies appear in *italics*.

AT+iSTCP:secure.sslserver.com,1503	Open a TCP/IP socket to a secure application.
<i>I/000</i>	iChip opens socket and returns handle 0.
AT+iSSL:0	iChip is instructed to negotiate an SSL3/TLS1 connection on socket

I/OK	handle 0. SSL3/TLS1 handshake was successful. SSL3/TLS1 connection established on socket handle 0.
AT+iSSND%:0,323:< ... 323 bytes of plaintext data ... >	Host sends 323 bytes of plain text data via SSL3/TLS1 socket. iChip will encrypt data and send cipher text over the Internet. The ‘%’ attribute indicates immediate flush.
I/OK	iChip encrypted and sent data.
AT+iRP4	Request socket status
I/(1267,-200,-200,-200,-200,-200,-200,-200,-200)	Socket 0 has 1267 plain text bytes buffered. The data was originally sent encrypted by the server. iChip decrypted the cipher text in the background.
AT+iSRCV:0	Command to retrieve buffered plain text.
I/1267:< ... 1267 bytes of plaintext data ... >	iChip transmits buffered data to host.
AT+iscls:0	Close socket handle 0
I/OK	SSL3/TLS1 socket is closed
I/DONE	iChip offline

Secure Socket Protocol Parameters

+iCS — Define the SSL3/TLS1 Cipher Suite

Syntax:	AT+iCS= <i>n</i>
	Set the cipher suite to be used in SSL3/TLS1 negotiations with a secure server.
	The default value '0' is the all-cipher selection. With this value, iChip sends its full list of supported ciphers to the server. The server selects the most appropriate cipher to use during the handshake procedure. When a specific value is specified, iChip requires the server to use that specific cipher.
Parameters:	<i>n</i> = A supported cipher suite code, as defined in RFC2246.
Command Options:	
<i>n</i> = 0	Set cipher suite to 'propose all'. When CS is set to 'propose all', iChip offers all supported cipher suites for SSL3/TLS1 negotiations. The server selects the most appropriate cipher suite during the handshake procedure.
<i>n</i> = 4	Set cipher suite to SSL_RSA_WITH_RC4_128_MD5
<i>n</i> = 5	Set cipher suite to SSL_RSA_WITH_RC4_128_SHA
<i>n</i> = 10	Set cipher suite to SSL_RSA_WITH_3DES_EDE_CBC_SHA
<i>n</i> = 47	Set cipher suite to TLS_RSA_WITH_AES_128_CBC_SHA
<i>n</i> = 53	Set cipher suite to TLS_RSA_WITH_AES_256_CBC_SHA
Default:	0 (Propose All)
Result code:	
I/OK	If <i>n</i> is a supported cipher suite code
I/ERROR	Otherwise
AT+iCS?	Returns the current cipher suite value followed by I/OK.
AT+iCS=?	Returns the message "0,4,5,10,47,53" followed by I/OK.

+iCA — Define SSL3/TLS1 Certificate Authority

Syntax:	AT+iCA= <i>tca</i> Set the certificate of the trusted certificate authority. This authority is the one eligible to sign a server's certificate. iChip accepts a server's identity only if its certificate is signed by this authority.
Parameters:	<i>tca</i> = PEM format DER-encoded X509 certificate
Command Options:	
<i>tca</i> =<CR><CR>	Empty: No trusted certificate authority.
<i>tca</i> =<cert>	<i>cert</i> is referenced as the trusted certificate authority's certificate during SSL3/TLS1 socket connection establishment (handshake). iChip establishes an SSL3/TLS1 socket connection only to servers having a certificate authenticated by this certificate authority. iChip expects <i>cert</i> to be multiple lines separated by <CR>, beginning with -----BEGIN CERTIFICATE----- and terminating with -----END CERTIFICATE-----.
Default:	Empty. No trusted Certificate Authority defined.
Result code:	
I/OK	If <i>tca</i> is an empty or legal certificate.
I/ERROR	Otherwise
AT+iCA?	Report the current trusted certificate contents. The reported value displays the Certificate Authority name, certificate validity date range, and the entire PEM contents. If the trusted certificate is empty, only <CRLF> is returned. The reply is followed by I/OK.
AT+iCA=?	Returns the message ' String ' followed by I/OK. Sample PEM format DER-encoded X509 certificate: -----BEGIN CERTIFICATE----- MIICPDCCAaUCEHC65B0Q2Sk0tjjKewPMur8wDQYJKo ZIhvcNAQECBQAwXzELMAkG A1UEBhMCMVVMxZzAVBgNVBAoTDIzlcm1lTaWduLCBJ bmMuMTcwNQYDVQQLEy5DbGFz cyAzIFB1Ym90YyBQcm1lYXJ5IENlcnRpb24g QXV0aG9yaXR5MB4XDTE2MDUwMDEyMjE0MDgW MDEyOTAwMDAwMFOxDTI4MDgWMTIzNTk1OVowX

zELMAkGA1UEBhMCMVVMxFzAVBgNV

BAoTDIZlcm1TaWduLCBJbmMuMTcwNQYDVQQLEy5
DbGFzcyAzIFB1Ym91bnR1eSBBQmlt

YXJ5IENlcnRpb24gQXV0aG9yaXR5MIGfMA0
GCSqGSIb3DQEBAQUAA4GN

ADCBiQKBgQDJXFme8huKARS0EN8EQNvjV69qRUCP
hAwL0TPZ2RHP7gJYHyX3KqhE

BarsAx94f56TuZoAqiN91qyFomNFx3InzPRMxnVx0jnvT0
Lwdd8KkMaOIG+YD/is

I19wKTakyYbnsZogy1Olhec9vn2a/iRFM9x2Fe0PonFkTG
UugWhFpwIDAQABMA0G

CSqGSIb3DQEBAgUAA4GBALtMEivPLCYATxQT3ab7/
AoRhIzzKBxki98tsX63/Do

lbwdj2wsqFHM9ikwFPwTtYmwHYBV4GSXiHx0bH/59A
hWM1pF+NEHJwZRDmJXNyc

AA9WjQKZ7aKQRUzkuxCkPfAyAw7xzvjoyVGM5mKf5
p/AfbdynMk2OmufTqj/ZA1k

-----END CERTIFICATE-----

+iCERT — Define SSL3/TLS1 Certificate

Syntax:	AT+iCERT= <i>ct</i> Set iChip's SSL3/TLS1 certificate. Some SSL3/TLS1 servers require the client side to authenticate its identity by requesting the client to provide a certificate during the SSL socket negotiation phase. This is called "client side authentication". If the CERT parameter contains a certificate, iChip provides it to the server upon request. iChip also needs a private key (see PKEY parameter) in order to encrypt its certificate before sending it to the server. In addition, the certificate should be signed by a certificate authority accepted by the server for the client side authentication to succeed.
Parameters:	<i>ct</i> = PEM format DER-encoded X509 Certificate
Command Options:	
<i>ct</i> = <CR><CR>	Empty. No trusted certificate authority.
<i>ct</i> =< <i>cert</i> >	<i>cert</i> is used as iChip's certificate during client side authentication. The certificate must be signed by a certificate authority acceptable by the server. iChip expects <i>cert</i> to be multiple lines separated by <CR>, beginning with -----BEGIN CERTIFICATE----- and terminating with -----END CERTIFICATE-----.
Default:	Empty. No trusted certificate authority defined.
Result code:	
I/OK	If <i>ct</i> is an empty or legal certificate.
I/ERROR	Otherwise
AT+iCERT?	Displays current certificate contents. If the trusted certificate is empty, only <CRLF> is returned, followed by I/OK.
AT+iCERT=?	Returns the message ' String ' followed by I/OK.

+iPKEY — Define iChip's Private Key

Syntax:	AT+iPKEY= <i>pky</i> Set iChip's private key. The private key is required to perform an RSA encryption of its certificate (see CERT parameter) when performing client side authentication. Special care should be taken to protect private key contents from unauthorized parties. For this reason, once the private key is stored on iChip, it cannot be read – only erased or overwritten.
Parameters:	<i>pky</i> = PEM format
Command Options:	
<i>pky</i> =<CR><CR>	Empty. Any existing private key is erased.
<i>pky</i> =< <i>pkey</i> >	<i>pkey</i> is used as iChip's private key to RSA encrypt its certificate during client side authentication. iChip expects <i>pkey</i> to be multiple lines separated by <CR>, beginning with -----BEGIN RSA PRIVATE KEY----- and terminating with -----END RSA PRIVATE KEY-----
Default:	Empty. No private key defined.
Result code:	
I/OK	If <i>pky</i> is an empty or legal private key.
I/ERROR	Otherwise
AT+iPKEY?	Reports the current private key's strength (number of bits in key). If the key is empty, only <CRLF> is returned. There is no way to retrieve <i>pkey</i> contents. The reply is followed by I/OK.
AT+iPKEY=?	Returns the message ' String ' followed by I/OK.
Example:	-----BEGIN RSA PRIVATE KEY----- MIICXAIBAAKBgQCoMGVcZ3HNFB/cRfWP7vdZr RK+YB+lez07mAN6Zcd4C19Xi6M6 dmewb6qQ6TRYC1gBhJ+KtMopGoqQ3v1VSu0Ve/Zrj WNxLN9UAAtRMubtkGz2j6OCt Ix4WsFUWebF8QEEem9+3coMnRqtAdluYEU2F2PTe WUsQfjRQQmBjus/y0wwIDAQAB AoGBAKWaKWOHk1zbENfhpn1XTQNmT4tVuDNH Gi6gaeRNbM79W54mpsy8ozHtcWOH

y3tZiAjOngyEIH3CXWdxuL0PrkmdSk39+V0EIuA0sR
xyUTb3/LIDU9DpxlYXBYK5
Kclq2qH5GBv28QJChG6/dfvuO8a1JyPwD61iOvBvBy
e/C7QRAkEA1uU7pT8ejcxf
ZLwaBwUift9Y1kpzrdHYnqJggrhGeZq4bIb8ioOFEgB
+JKXSxaQZgxUsIkDVzkO/
+J/H8KZKywJBAMhcGEftwPqtZMWyqis7rSUpsewax
g79QYDZVSRwi5ynLqtqui4d
GVsftbXvtZHRs8uyp3plTFUVFvPRsUJpukCQEZYJz
dola+OS8dOEooymLhWp1y4
U2ur2wNF37V6iz/aBJMvPSJ7MuhP2QpSgeHghax/CF
TCRFS1yPzMBFNTcDkCQEHq
ko5veNK/4uxruDJbAr68Ne3gbRKXXUp/tdQ0NqpGEk
OQ7EmphyDhHk4J2+lqXUWB
tDm/Q9qmAmyfJ8BBSakCQAaO10MGdUnyFuanp19j
RfLB29oOqMQqyV90r25AxOcN
HD8Jsmn5vBYm4wdtR8x84Gh7128RfuBS8J0hFb90y
RY=
-----END RSA PRIVATE KEY-----

Known Bugs

Bug Description: When attempting to retrieve a list of files from a directory stored on an FTP server using the +iFDNL command, iChip returns **I/OK** and **I/ONLINE**, as if that directory is empty.

Workaround: The command AT+iFDL (FTP Directory Listing) can be used to cover most functions performed by +iFDNL.

Solution: Firmware version dealing correctly with +iFDNL will be available soon.

Bug Description: When attempting to force iChip into SerialNET mode using either the AT+i@SNMD or AT+i!SNMD command, iChip returns **I/OFFLINE**.

Workaround: You can still use the AT+iSNMD command (without any flags) to force iChip into SerialNET mode.

Solution: Firmware version dealing correctly with the AT+i@SNMD and AT+i!SNMD commands will be available soon.

Bug Description: Sending and receiving of files of 2Kb or more over a UDP socket fails.

Solution: Firmware version supporting sending and receiving of files of any size over a UDP socket will be available soon.

Bug Description: After opening a secure FTP link, iChip fails to list the files in a directory on the FTP server.

Bug Description: The **AT+iCS=?** command returns error no. (042) *Illegal value*, instead of “0, 4, 5, 10, 47, 53”.

Bug Description: Following the **AT+iFD** command, iChip requires a software reset to get an IP address.

Bug Description: