

CipherLab **User Guide**

BASIC Language Programming Part II: Data Communications

For 8 Series Mobile Computers

Version 5.06



Copyright © 2007~2015 CIPHERLAB CO., LTD.
All rights reserved

The software contains proprietary information of CIPHERLAB CO., LTD.; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited.

Due to continued product development this information may change without notice. The information and intellectual property contained herein is confidential between CIPHERLAB and the client and remains the exclusive property of CIPHERLAB CO., LTD. If you find any problems in the documentation, please report them to us in writing. CIPHERLAB does not warrant that this document is error-free.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of CIPHERLAB CO., LTD.

For product consultancy and technical support, please contact your local sales representative. Also, you may visit our web site for more information.

The CipherLab logo is a registered trademark of CIPHERLAB CO., LTD.

All brand, product and service, and trademark names are the property of their registered owners.

The editorial use of these names is for identification as well as to the benefit of the owners, with no intention of infringement.

CIPHERLAB CO., LTD.

Website: <http://www.cipherlab.com>

RELEASE NOTES

Version	Date	Notes
5.06	Jun. 22, 2015	<p>Part I</p> <ul style="list-style-type: none">▶ Modified: Appendix I – SYMBOLOGY PARAMETER TABLE FOR CCD/LASER/LONG RANGE READER:<ul style="list-style-type: none">: 300~306, 308 updated with 8400: 312~317 added for Quiet Zone check settings (8200/8400)▶ Modified: Appendix II – SCAN ENGINE, CCD OR LASER:<ul style="list-style-type: none">: 300~306 updated with 8400: 308 updated with 8400 (Addon Security): 312 ~ 317 added for Quiet Zone check settings (8200/8400) <p>Part II</p> <ul style="list-style-type: none">▶ Modified: Appendix III – Wireless Networking table updated
5.05	Mar. 06, 2015	<p>Part I</p> <ul style="list-style-type: none">▶ Modified: Appendix I – update “Symbology Parameter Table for CCD/Laser/Long Range Reader” with 180~299 & 300~308▶ Modified: Appendix II – Scan Engine, CCD or Laser – UPC/EAN Families: “EAN-13 Addon Mode” and “Addon Security for UPC/EAN” added <p>Part II</p> <p>- None –</p>
5.04	Mar. 28, 2014	<p>Part I</p> <ul style="list-style-type: none">▶ Modified: Appendix I - SYMBOLOGY PARAMETER TABLE I<ul style="list-style-type: none">>No. (N1%): 87 (GTIN -> GTIN-14)▶ Removed: Appendix I - SYMBOLOGY PARAMETER TABLE II<ul style="list-style-type: none">>No. (N1%): 188 (GS1 formatting for GS1 DataMatrix)▶ Modified: Appendix II Symbology Parameters – Scan Engine, CCD or Laser<ul style="list-style-type: none">>UPC/EAN FAMILIES: No. 87 (GTIN -> GTIN-14)▶ Removed: Appendix II – Scan Engine, CCD or Laser – 2D SCAN ENGINE ONLY<ul style="list-style-type: none">>2D SYMBOLOGIES MAXICODE, DATA MATRIX & QR CODE: No. 188 <p>Part II</p> <ul style="list-style-type: none">▶ Modified: 3.1.2 - Commands<ul style="list-style-type: none">>“A\$” variable table updated for SET_NET_PARAMETER▶ Modified: Appendix II – Net Parameters by Index – Wireless Networking<ul style="list-style-type: none">>-92~-96 (GET)/92~96 (SET) indexes updated

- ▶ Replace “RSS” with “GS1 DataBar”
- ▶ Modified: **Chapter 1** – Windows 95/98/7 supported (chapter 1, 2)
- ▶ Modified: **2.3 Configure Menu** – descriptions for “Create DBF Files” command revised
- ▶ Modified: **4.15 KEYPAD COMMANDS | 4.15.1 GENERAL** –
 - >8000 supports OSK_TOGGLE, GET_TRIGGER, SET_TRIGGER, SET_PWR_KEY commands
 - >SET_MIDDLE_ENTER command added for 8400/8700
 - >SET_PISTOL_ENTER command added for 8200/8700
- ▶ Modified: **4.18 Fonts | 4.18.4 Special Font Files** –
 - >Turkey (33) added to GET_LANGUAGE, SET_LANGUAGE
- ▶ Modified: **Appendix I - SYMBOLOGY PARAMETER TABLE I**
 - >No. (N1%): 54, 173, 174, 175, 176, 177, 178, 179 added
- ▶ Modified: **Appendix I - SYMBOLOGY PARAMETER TABLE II**
 - >No. (N1%): 94 (Disable TCIF Linked Code 39 by default)
 - >No. (N1%): 174, 176 ~ 179/181 ~ 188 added
- ▶ Modified: **Appendix II Symbology Parameters – Scan Engine, CCD or Laser**
 - >Code39: No. 173
 - >CODE 128/EAN-128/ISBT 128: No. 174
 - >GS1 DataBar FAMILY: No. 175
 - >UPC/EAN FAMILIES: No. 54
 - >UPC/EAN FAMILIES: UPC-E Triple Check descriptions
- SCAN ENGINE, 2D OR (EXTRA) LONG RANGE LASER**
 - >CODE 128 | UCC/EAN-128: No. 174
 - >GS1 DataBar FAMILY: No. 183~185
- 2D SCAN ENGINE ONLY**
 - >COMPOSITE CODES | CC-A/B/C: No. 186~187
 - >TLC-39: No. 94 (Disable TCIF Linked Code 39 by default)
 - >2D SYMBOLOGIES|MAXICODE, DATA MATRIX & QR CODE: No. 188
- ▶ Modified: **Appendix III Scanner Parameter –**
 - >READ REDUNDANCY: No. 182
 - >USER PREFERENCES: No. 181

Part II

- None –

5.02	Mar. 27, 2013	Part I	<ul style="list-style-type: none"> ▶ Modified: 4.7.2 Code Type – CodeType Table II: add 8400/8700 2D scan engine to Composite_CC_A/B/C symbologies (Decimal 47/55/118) ▶ Modified: 4.15.1 General – OSK_TOGGLE, SET_PWR_KEY: support for 8400/8700 added ▶ Modified: Appendix I – Symbology Parameter Table II: add 8400/8700 2D scan engine to No. 44 (Convert UPC-A to EAN-13) ▶ Modified: Appendix II – Scan Engine, 2D or (extra) Long Range Laser: add 8400/8700 2D scan engine to No. 44 (Convert UPC-A to EAN-13)
		Part II	- None –
5.01	Dec. 07, 2012	Part I	<ul style="list-style-type: none"> ▶ Modified: 1.1 Directory Structure – Font Files (8200/8400/8700) ▶ Modified: 4.7.2 Code Type Table II – Symbology added (No. 47/55/118) ▶ Added: 4.15 Keypad Commands – OSK_TOGGLE, SET_PWR_KEY commands added ▶ Modified: 4.15.2 ALPHA KEY – GET_ALPHA_STATE command removed ▶ Modified: 4.18.1 Font Size – 20x20 added ▶ Modified: 4.18.4 – SELECT_FONT command modified ▶ Modified: Appendix I – Symbology Parameter Table II – value & description added (No. 44)
		Part II	- None –
4.24	Oct. 23, 2012	Part I	<ul style="list-style-type: none"> ▶ Modified: 4.15 Keypad Commands 4.15.1 General – SET_TRIGGER, GET_TRIGGER commands added for 8200/8400/8700; CHECK_ENTER_KEY for 8200/8700; SET_MIDDLE_ENTER for 8200 ▶ Modified: 4.16 LCD Commands 4.16.1 Properties – BACKLIT command revised; GET_BKLIT_LEVEL, SET_AUTO_BKLIT, SET_BKLIT_LEVEL commands added for 8200/8400/8700 ▶ Modified: 2.5 Help Menu one command is provided (not three) ▶ Modified: Appendix VII Key Code Table – MCR/LCR/RCR added for 8200; LCR/RCR added for 8700
		Part II	<ul style="list-style-type: none"> ▶ New: 4.3 Scanning for Wi-Fi Hotspots – WIFI_SCAN command added for 8200/8400/8700
4.23	July. 02, 2012	Part II	<ul style="list-style-type: none"> ▶ New: Appendix II add Wi-Fi Profile index ▶ New: Appendix IV add PCAT — Swiss(German) and Hungarian for 8400/8700. ▶ New: 4.2 Wi-Fi Profile

4.22	Apr. 26, 2012	Part II <ul style="list-style-type: none"> ▶ Add PCAT — Swiss(German) and Hungarian for 8200. ▶ Task: Rename FTP Files — add File already exists.
4.21	Mar. 13, 2012	Part I <ul style="list-style-type: none"> ▶ Modified: Appendix I ScannerDesTbl Array Symbology Table II Note: MSI and Code 11 are disabled for 8400 2D scan engine by default. ▶ Modified: Appendix II Symbology Parameters Scan Engine, 2D or (Extra) Long Range Laser – Note: MSI and Code 11 are disabled for 8400 2D scan engine by default. Part II <ul style="list-style-type: none"> ▶ New: 10.3.1 "Command: FTP_ROUTINE\$" Remarks FTP Task Variable Table - Note (4)
4.20	Dec. 12, 2011	Part I <ul style="list-style-type: none"> ▶ None Part II <ul style="list-style-type: none"> ▶ Modified: 8780 removed from the manual. ▶ Modified: 10.3.1: Parameters to rename / delete FTP files added to command FTP_ROUTINE\$ for 8200 & 8400. ▶ Modified: Appendix V: FTP messages for renaming / deleting FTP files added.
4.10	Jul. 07, 2011	Part I <ul style="list-style-type: none"> ▶ Modified: 4.19 Memory Commands — 8700's updated Part II <ul style="list-style-type: none"> ▶ Modified: 5.1 Bluetooth Profiles Supported — Bluetooth HSP for 8200 removed ▶ Modified: Appendix IV Examples — Bluetooth HSP (8200 Only) removed
4.00	Mar. 21, 2011	BASIC Programming Guide split into Part I: Basics and Hardware Control, and Part II: Data Communications <ul style="list-style-type: none"> ▶ Modified: add 8200 support ▶ Modified: add 8700 support ▶ Modified: remove 8580/8590

Part I

- ▶ 3.2.1 Variable Names and Declaration Characters — add “About Real Number”
- ▶ 4.6.2 System Information — SYSTEM_INFORMATION\$() for 8200 bootloader version
- ▶ 4.10 Buzzer Commands — BEEP() allows setting 8200’s speaker mute
- ▶ 4.15.3 FN Key — Auto Resume mode for 8300 allows re-pressing the function key to exit the function mode
- ▶ Appendix VII Key Code Table — updated for 8200/8700

Part II

- ▶ Add support of Bluetooth HSP and FTP for 8200
- ▶ 1.3.1 Commands — SET_COM_TYPE() supports USB Virtual COM_CDC and Bluetooth HSP for 8200
- ▶ 8.1.2 USB Virtual COM — add support of USB Virtual COM_CDC for 8200
- ▶ 9 GPS Functionality — add support of GPS for 8700
- ▶ 10 FTP Functionality

CONTENTS

RELEASE NOTES	- 3 -
INTRODUCTION.....	1
COMMUNICATION PORTS.....	3
1.1 Basics	4
1.1.1 Communication Parameters	4
1.1.2 Receive & Transmit Buffers	4
1.2 Flow Control.....	5
1.2.1 RTS/CTS.....	5
1.2.2 XON/XOFF.....	6
1.2.3 Commands	7
1.3 Configure Settings.....	8
1.3.1 Commands	8
1.4 Open and Close COM.....	12
1.4.1 Commands	12
1.5 Read and Write Data	13
1.5.1 Commands	13
TCP/IP COMMUNICATIONS.....	15
2.1 Configure Settings.....	16
2.1.1 Commands	16
2.2 Open and Close Connection.....	17
2.2.1 Commands	17
2.3 Read and Write Data	19
2.3.1 Commands	19
2.4 Get TCP/IP Message	21
2.4.1 Command	21
2.5 Get TCP/IP Error Code.....	22
2.5.1 Command	22
WIRELESS NETWORKING.....	25
3.1 Network Configuration.....	26
3.1.1 Implementation.....	26
3.1.2 Commands	26
3.2 Initialization & Termination	29
3.2.1 Overview.....	29
3.2.2 Commands	31
3.3 Network Status.....	32
3.3.1 Command	32

IEEE 802.11B/G/N	33
4.1 Obsolete Command	34
4.2 Wi-Fi Profile.....	34
4.2.1 Commands	34
4.3 Scanning for Wi-Fi Hotspots.....	38
BLUETOOTH.....	39
5.1 Bluetooth Profiles Supported	40
5.2 Frequent Device List.....	41
5.3 Inquiry	43
5.3.1 Command	43
5.4 Pairing.....	44
5.4.1 Command	44
GSM/GPRS	45
6.1 Data Format.....	46
6.2 Security	49
6.2.1 PIN Procedure	49
6.2.2 PUK Procedure	50
6.3 PIN Code	51
6.3.1 Commands	51
MODEM, ETHERNET & GPRS CONNECTION.....	53
7.1 PPP via Modem Cradle/RS-232	54
7.1.1 CONFIG Settings.....	54
7.2 Ethernet via Cradle	55
7.3 GPRS via Cradle	56
USB CONNECTION.....	57
8.1 Overview.....	58
8.1.1 USB HID	58
8.1.2 USB Virtual COM.....	58
8.1.3 USB Mass Storage Device	58
GPS FUNCTIONALITY	59
FTP FUNCTIONALITY	61
10.1 Configure Settings.....	62
10.1.1 Net Parameters by Index.....	62
10.1.2 Command: GET_NET_PARAMETER	62
10.1.3 Command: SET_NET_PARAMETER.....	62
10.2 Connect and Disconnect.....	63
10.2.1 Command: TCP_OPEN	63
10.2.2 Command: NClose.....	65
10.2.3 Delimiter Handling	66
10.3 Do FTP Task.....	68

10.3.1 Command: FTP_ROUTINE\$.....	69
10.4 Download Program Updates.....	71
10.4.1 Updating BASIC Runtime.....	71
10.4.2 Updating BASIC Application.....	73
10.4.3 Activating Programs.....	75
10.4.4 Command: UPDATE_BASIC	75
10.5 File Handling	77
10.5.1 DAT Files.....	77
10.5.2 DBF Files.....	79
10.6 SD Card Access.....	81
10.6.1 Directory.....	82
10.6.2 File Name	84
CRADLE COMMANDS	85
NET PARAMETERS BY INDEX.....	89
Wireless Networking.....	89
Bluetooth SPP, FTP, DUN.....	91
GSM/GPRS.....	92
PPP	93
USB	93
FTP.....	93
NET STATUS BY INDEX.....	95
Wireless Networking.....	95
Bluetooth SPP, FTP, DUN.....	97
GSM/GPRS.....	98
EXAMPLES	99
WLAN Example (802.11b/g)	99
WPA Enabled for Security	102
Bluetooth Examples.....	103
SPP Master	104
SPP Slave.....	105
Wedge Emulator via SPP.....	106
HID.....	108
DUN.....	111
DUN-GPRS.....	112
FTP (8200 Only).....	113
GSM/GPRS Examples	114
GPRS.....	114
GSM	115
USB Example.....	117
USB Virtual COM.....	117
USB HID.....	118
USB Mass Storage Device	120

FTP MESSAGE	121
Task: Connect.....	121
Task: Get Directory	121
Task: Change Directory	122
Task: Upload File	122
Task: Append to File	122
Task: Download File	122
Task: Rename FTP Files.....	122
Task: Delete FTP Files	123
INDEX.....	125

INTRODUCTION

CipherLab BASIC Compiler provides users with a complete programming environment to develop application programs for CipherLab 8 Series Mobile Computers using the BASIC language. The Windows-based Basic Compiler comes with a menu-driven interface to simplify software development and code modifications. Many system configurations, such as COM port properties and database file settings can be set up in the menus. Using this powerful programming tool to get rid of lengthy coding, users can develop an application to meet their own needs efficiently. The CipherLab BASIC Compiler has been modified and improved since its first release in November 1997. Users can refer to RELEASE.TXT for detailed revision history.

This manual is meant to provide detailed information about how to use the BASIC Compiler to write application programs for CipherLab 8 Series Mobile Computers. It is organized in chapters giving outlines as follows:

Part I: Basics and Hardware Control

- | | |
|-----------|---|
| Chapter 1 | "Development Environment" – gives a concise introduction about the CipherLab BASIC Compiler, the development flow for applications, and the BASIC Compiler Run-time Engines. |
| Chapter 2 | "Using CipherLab BASIC Compiler" – gives a tour of the programming environment of the BASIC Compiler. |
| Chapter 3 | "Basics of CipherLab BASIC Language" – discusses the specific characteristics of the CipherLab BASIC Language. |
| Chapter 4 | "BASIC Commands" – discusses all the supported BASIC functions and statements. More than 200 BASIC functions and statements are categorized according to their functions, and discussed in details. |

Part II: Data Communications

- | | |
|------------|-------------------------------------|
| Chapter 1 | "Communication Ports" |
| Chapter 2 | "TCP/IP Communications" |
| Chapter 3 | "Wireless Networking" |
| Chapter 4 | "IEEE 802.11b/g" |
| Chapter 5 | "Bluetooth" |
| Chapter 6 | "GSM/GPRS" |
| Chapter 7 | "Modem, Ethernet & GPRS Connection" |
| Chapter 8 | "USB Connection" |
| Chapter 9 | "GPS Functionality" |
| Chapter 10 | "FTP Functionality" |

COMMUNICATION PORTS

There are at least two communication (COM) ports on each mobile computer, namely *COM1* and *COM2*. The user has to call **SET_COM_TYPE** to set up the communication type for the COM ports before using them. Commands for triggering the COM event: **OFF COM**, **ON COM GOSUB...**

Note: SET_COM_TYPE is not applicable to RFID (COM 4).

The table below shows the mapping of the communication (COM) ports. Specifying which type of interface is to be used, the user can use the same commands to open, close, read, and write data (**OPEN_COM**, **CLOSE_COM**, **READ_COM\$**, and **WRITE_COM**).

Series	COM1	COM2	COM3	COM4	COM5
8000	Serial IR, IrDA	Acoustic Coupler, Bluetooth	N/A	N/A	N/A
8200	RS-232	Bluetooth	N/A	N/A	USB
8300	RS-232, Serial IR, IrDA	Acoustic Coupler, RF, Bluetooth	N/A	RFID	N/A
8400	RS-232	Bluetooth	N/A	N/A	USB
8500	Serial IR, IrDA	Bluetooth	GSM	RFID	N/A
8700	RS-232	Bluetooth	3.5G	RFID	USB

Note: (1) The Bluetooth profiles supported include SPP, DUN, HID, and FTP.
(2) Bluetooth FTP is supported on 8200 only.
(3) GSM/GPRS/EDGE or UMTS/HSDPA services are supported on 8700.

IN THIS CHAPTER

1.1 Basics.....	4
1.2 Flow Control	5
1.3 Configure Settings	8
1.4 Open and Close COM	12
1.5 Read and Write Data	13

1.1 BASICS

1.1.1 COMMUNICATION PARAMETERS

RS-232 Parameters	
Baud Rate:	115200, 76800, 57600, 38400, 19200, 9600, 4800, 2400
Data Bits:	7 or 8
Parity:	Even, Odd, or None
Stop Bit:	1
Flow Control:	RTS/CTS, XON/XOFF, or None
Serial IR Parameters	
Baud Rate:	115200, 57600, 38400, 19200, 9600
Data Bits:	8
Parity:	Even, Odd, or None
Stop Bit:	1
Flow Control:	None
IrDA, USB Parameters	
Baud Rate:	Ignored, included only for compatibility in coding.
Data Bits:	Ignored, included only for compatibility in coding.
Parity:	Ignored, included only for compatibility in coding.
Stop Bit:	Ignored, included only for compatibility in coding.
Flow Control:	Ignored, included only for compatibility in coding.

1.1.2 RECEIVE & TRANSMIT BUFFERS

Receive Buffer

A 256 byte FIFO buffer is allocated for each port. The data successfully received is stored in this buffer sequentially (if any error occurs, e. g. framing, parity error, etc., the data is simply discarded). However, if the buffer is already full, the incoming data will be discarded and an overrun flag is set to indicate this error.

Transmit Buffer

The system does not allocate any transmit buffer. It simply records the pointer of the string to be sent. The transmission stops when a null character (0x00) is encountered. The application program must allocate its own transmit buffer and not to modify it during transmission.

1.2 FLOW CONTROL

To avoid data loss, three options of flow control are supported and done by background routines.

- 1) None (= Flow control is disabled.)
- 2) RTS/CTS
- 3) XON/XOFF

Note: Flow control is only applicable to the direct RS-232 COM port, which is usually assigned as COM1.

1.2.1 RTS/CTS

RTS now stands for *Ready for Receiving* instead of *Request To Send*, while CTS for *Clear To Send*. The two signals are used for hardware flow control.

Receive

The RTS signal is used to indicate whether the storage of receive buffer is free or not. If the receive buffer cannot take more than 5 characters, the RTS signal is de-asserted, and it instructs the sending device to halt the transmission. When its receive buffer becomes enough for more than 15 characters, the RTS signal becomes asserted again, and it instructs the sending device to resume transmission. As long as the buffer is sufficient (may be between 5 to 15 characters), the received data can be stored even though the RTS signal has just been negated.

Transmit

Transmission is allowed only when the CTS signal is asserted. If the CTS signal is negated (= de-asserted) and later becomes asserted again, the transmission is automatically resumed by background routines. However, due to the UART design (on-chip temporary transmission buffer), up to five characters might be sent after the CTS signal is de-asserted.

1.2.2 XON/XOFF

Instead of using RTS/CTS signals, two special characters are used for software flow control — XON (hex 11) and XOFF (hex 13). XON is used to enable transmission while XOFF to disable transmission.

Receive

The received characters are examined to see if it is normal data (which will be stored to the receive buffer) or a flow control code (set/reset transmission flag but not stored). If the receive buffer cannot take more than 5 characters, an XOFF control code is sent. When the receive buffer becomes enough for more than 15 characters, an XON control code will be sent so that the transmission will be resumed. As long as the buffer is sufficient (may be between 5 to 15 characters), the received data can be stored even when in XOFF state.

Transmit

When the port is opened, the transmission is enabled. Then every character received is examined to see if it is normal data or flow control codes. If an XOFF is received, transmission is halted. It is resumed later when XON is received. Just like the RTS/CTS control, up to two characters might be sent after an XOFF is received.

Note: If receiving and transmitting are concurrently in operation, the XON/XOFF control codes might be inserted into normal transmit data string. When using this method, make sure that both sides feature the same control methodology; otherwise, dead lock might happen.

1.2.3 COMMANDS

GET_CTS **8200, 8300, 8400, 8700**

Purpose To check the current CTS state on the direct RS-232 port.

Syntax `A% = GET_CTS(N%)`

Remarks "A%" is an integer variable to be assigned to the result.

A%	Meaning
0	CTS signal is negated. (= space)
1	CTS signal is asserted. (= mark)

"N%" is an integer variable, indicating on which COM port to get CTS level.

Example `A% = GET_CTS(1)`

SET_RTS **8200, 8300, 8400, 8700**

Purpose To set the RTS signal on the direct RS-232 port.

Syntax `SET_RTS(N1%, N2%)`

Remarks "N1%" is an integer variable, indicating on which COM port to set RTS level.

"N2%" is an integer variable, indicating the RTS state.

N2%	Meaning
0	RTS signal is negated. (= space)
1	RTS signal is asserted. (= mark)

Example `SET_RTS(1, 1)` ' set COM1 RTS to the "mark" state

1.3 CONFIGURE SETTINGS

1.3.1 COMMANDS

COM_DELIMITER

Purpose	To change delimiter of sending and receiving string for a specified COM port.	
Syntax	COM_DELIMITER(<i>N%</i> , <i>C%</i>)	
Remarks	<p>The default COM_DELIMITER is 0x0d (CR).</p> <p>"<i>N%</i>" is an integer variable, indicating which COM port is to be set.</p> <p>"<i>C%</i>" is an integer variable, representing the ASCII code of the delimiter character, in the range of 0 to 255. If it is negative, no delimiter will be applied.</p>	
Example	COM_DELIMITER(1, 13)	' use RETURN as delimiter
	COM_DELIMITER(1, 10)	' use Line Feed as delimiter

IRDA_TIMEOUT

8000, 8300, 8500

Purpose	To set the timeout for IrDA connection.	
Syntax	IRDA_TIMEOUT(<i>N%</i>)	
Remarks	<p>"<i>N%</i>" is an integer variable in the range of 1 to 8, indicating a specified period of time.</p>	

N%	Meaning
1	3 sec
2	8 sec
3	12 sec
4	16 sec
5	20 sec
6	25 sec
7	30 sec
8	40 sec

Example	IRDA_TIMEOUT(7)	' set timeout to 30 seconds
---------	-----------------	-----------------------------

SET_COM_TYPE

Purpose To assign the communication type to a specified COM port.

Syntax SET_COM_TYPE(*N%*, *type%*)

Remarks "*N%*" is an integer variable, indicating which COM port is to be set. Refer to the COM Port Mapping table.

"*type%*" is an integer variable, indicating the type of interface.

TYPE%	Meaning
1	Direct RS-232
2	I/O pins via cradle (8200/8400/8700)
3	Serial IR via IR transceiver (8000/8300/8500); Auto-detect (8200/8400/8700)
4	Standard IrDA (8000/8300/8500)
5	RF, Bluetooth SPP/DUN/HID RF, Bluetooth SPP/DUN/HID (8200)
6	GSM_SMS (8500/8700)
7	Acoustic Coupler (8000/8300); GSM_Modem (8500/8700)
8	USB HID (8400/8700)
9	USB Virtual COM (8200/8400/8700)
10	USB Mass Storage (8200/8400/8700)
11	USB Virtual COM_CDC (8200/8700)

This function needs to be called BEFORE opening a COM port. However, it is not necessary for RFID.

Note that the COM port mapping is different for each model of mobile computer, and a COM port may not support all the communication types.

Example SET_COM_TYPE(1, 3) ' set COM1 of 8300 to serial IR communication

SET_COM

Purpose	To set parameters for a specified COM port.
Syntax	SET_COM(<i>N%</i> , <i>Baudrate%</i> , <i>Parity%</i> , <i>Data%</i> , <i>Handshake%</i>)
Remarks	<p>This command needs to be called BEFORE opening a COM port. However, it is not necessary for RF, GSM, and RFID.</p> <p>This command also serves Bluetooth configuration for SPP, DUN, HID and Wedge. Refer to Bluetooth Examples.</p>

Parameters	Values	Remarks
<i>N%</i>	1 or 2	Indicates which COM port is to be set.
<i>Baudrate%</i>	1: 115200 bps 2: 76800 bps ^{Note} 3: 57600 bps 4: 38400 bps 5: 19200 bps 6: 9600 bps 7: 4800 bps ^{Note} 8: 2400 bps ^{Note}	Specifies the baud rate of the COM port. ▶ For acoustic coupler setting, 1 ~ 4 is used to indicate its volume (low to high).
	Note: These are not applicable to Serial IR.	
<i>Parity%</i>	1: None 2: Odd 3: Even 4: Cradle commands	Specifies the parity of the COM port. Refer to Appendix IV — Cradle Commands.
<i>Data%</i>	1: 7 data bits 2: 8 data bits	Specifies the data bits of the COM port.
<i>Handshake%</i>	1: None 2: CTS/RTS 3: XON/XOFF 4: Wedge 5 ~ 14: 2 stop bits 15: for DTMF only	▶ Specifies the method of flow control for direct RS-232. ▶ Set 1 for Serial IR or IrDA. ▶ For 8000/8300 Series, to disable Reserved Host Commands, add 16 to the value of Handshake%. (see table below) ▶ For 8300 Series, to enable URPower 5V, add 32 to the value of Handshake%. (see table below) ▶ For acoustic coupler, 5 ~ 14 is used to indicate the character delay for 2 stop bits; 15 is for DTMF mode only. 5 for Bell202 – 0 character delay; 6 for Bell202 – 1 character delay; 7 for Bell202 – 3 characters delay;

		8 for Bell202 – 5 characters delay; 9 for Bell202 – 10 characters delay; 10 for V23 – 0 character delay; 11 for V23 – 1 character delay; 12 for V23 – 3 characters delay; 13 for V23 – 5 characters delay; 14 for V23 – 10 characters delay; 15 for DTMF 8 characters delay 2 characters gap.	
Value	Handshake%	Reserved Host Commands	URPower 5V
1	None	Enabled	Disabled
17	(=1+16)	Disabled	Disabled
33	(=1+32)	Enabled	Enabled
49	(=1+16+32)	Disabled	Enabled
2...	CTS/RTS	Same options	Same options
3...	XON/XOFF	Same options	Same options
4...	Wedge Emulator	Same options	Same options

Example

```

SET_COM(1, 1, 1, 2, 1)           ' COM1, 115200, None, 8, No handshake
SET_COM(1, 1, 1, 2, 17)          ' COM1, 115200, None, 8, No handshake,
                                   Reserved Host Commands disabled
SET_COM(1, 1, 1, 2, 33)          ' COM1, 115200, None, 8, No handshake,
                                   URPower 5V enabled
SET_COM(1, 1, 1, 2, 49)          ' COM1, 115200, None, 8, No
                                   handshake,Reserved Host Commands
                                   disabled, URPower 5V enabled

```

1.4 OPEN AND CLOSE COM

1.4.1 COMMANDS

CLOSE_COM

Purpose	To terminate communication and disable a specified COM port.
Syntax	CLOSE_COM(<i>N%</i>)
Remarks	" <i>N%</i> " is an integer variable, indicating which COM port is to be disabled.
Example	CLOSE_COM(2)

OPEN_COM

Purpose	To enable a specified COM port and initialize communication.
Syntax	OPEN_COM(<i>N%</i>)
Remarks	" <i>N%</i> " is an integer variable, indicating which COM port is to be enabled.
Example	OPEN_COM(1)

1.5 READ AND WRITE DATA

1.5.1 COMMANDS

IRDA_STATUS

8000, 8300, 8500

Purpose To check the IrDA connection status or transmission status.

Syntax A% = IRDA_STATUS(N%)

Remarks "A%" is an integer variable to be assigned to the result.

"N%" is an integer variable, indicating the action to take.

N%	Meaning
0	To check IrDA connection status. ▶ When A% = 0, it means the IrDA connection is disabled. ▶ When A% = 1, it means the IrDA connection is enabled.
1	To check whether data being transmitted successfully or not. ▶ A% is the length of string (delimiters are included).

Example

```
CLS
SET_COM_TYPE(1, 4)           ' set COM1 of 8300 as IrDA port
OPEN_COM(1)

Loop1:
IF (IRDA_STATUS(0) = 1) THEN  ' check connection
    BEEP(4400, 5)
ELSE
    GOTO Loop1
END IF
WRITE_COM(1, "My Data")
IF (IRDA_STATUS(1) = 7) THEN  ' check transmission
    PRINT "Write OK"
ELSE
    PRINT "Write NG"
END IF
CLOSE_COM(1)
```


READ_COM\$

Purpose	To read data from a specified COM port.
Syntax	<code>A\$ = READ_COM\$(N%)</code>
Remarks	<p>"A\$" is a string variable to be assigned to the data.</p> <p>"N%" is an integer variable, indicating from which COM port the data is to be read. If the receive buffer is empty, an empty string will be returned.</p>
Example	<pre> ON COM(1) GOSUB HostCommand ... HostCommand: Cmd\$ = READ_COM\$(1) CmdIdentifier\$ = LEFT\$(Cmd\$, 1) DBFNum% = VAL(MID\$(Cmd\$, 2, 1)) IDFNum% = VAL(MID\$(Cmd\$, 3, 1)) CardID\$ = RIGHT\$(Cmd\$, LEN(Cmd\$)-3) IF CmdIdentifier\$ = "-" THEN DEL_RECORD(DBFNum%, IDFNum%) ELSE ... </pre>

WRITE_COM

Purpose	To send a string to the host through a specified COM port.
Syntax	<code>WRITE_COM(N%, A\$)</code>
Remarks	<p>"N%" is an integer variable, indicating which COM port the data is to be sent to.</p> <p>"A\$" is a string variable, representing the string to be sent.</p>
Example	<pre> ON READER(1) GOSUB BcrData_1 ... BcrData_1: BEEP(2000, 5) Data\$ = GET_READER_DATA\$(1) WRITE_COM(1, Data\$) ... </pre>

TCP/IP COMMUNICATIONS

Here are the BASIC functions and statements related to TCP/IP networking.

Commands for triggering the TCPIP event: **OFF TCPIP, ON TCPIP GOSUB...**

IN THIS CHAPTER

2.1 Configure Settings	16
2.2 Open and Close Connection	17
2.3 Read and Write Data	19
2.4 Get TCP/IP Message	21
2.5 Get TCP/IP Error Code	22

2.1 CONFIGURE SETTINGS

2.1.1 COMMANDS

IP_CFG or IP_CONFIGURE

Purpose	To configure the TCP/IP setting.	
Syntax	IP_CFG(<i>index%</i> , <i>IP\$</i>) or IP_CONFIGURE(<i>index%</i> , <i>IP\$</i>)	
Remarks	This command is to be replaced by SET_NET_PARAMETER. Note that it is not necessary to configure the setting every time. " <i>index%</i> " is an integer variable, indicating a specific configuration item. " <i>IP\$</i> " is a string variable indicating the IP address that is to be configured.	
Example	<pre>IP_CFG(1, "192.168.1.241") ` set local IP IP_CFG(2, "255.255.255.0") ` set IP of subnet mask IP_CFG(3, "192.168.1.250") ` set IP of default gateway IP_CFG(4, "168.95.1.1") ` set IP of DNS server</pre>	

SOCKET_IP

Purpose	To get network settings.
Syntax	A\$ = SOCKET_IP(<i>port%</i>)
Remarks	This command is to be replaced by GET_NET_PARAMETER\$.
Example	NetSetting\$ = SOCKET_IP(0)

2.2 OPEN AND CLOSE CONNECTION

2.2.1 COMMANDS

DNS_RESOLVER

Purpose	To get the remote IP address by remote name.
Syntax	<i>IP\$</i> = DNS_RESOLVER(<i>A\$</i>)
Remarks	<p>"<i>IP\$</i>" is a string variable to be assigned to the result.</p> <p>"<i>A\$</i>" is a string variable, indicating a specific remote name.</p> <p>Note that it is necessary to define the DNS server IP before running this command.</p>
Example	GetIP\$ = DNS_RESOLVER("www.cipherlab.com")

NCLOSE

Purpose	To close a TCP/IP connection.
Syntax	NCLOSE(<i>N%</i>)
Remarks	" <i>N%</i> " is an integer variable in the range of 0 to 5, indicating the connection number.

N%	Meaning
0~3	TCP/IP connection
4	FTP connection (currently supported on 8000, 8200, 8300, 8400, 8700 only)
5	Bluetooth FTP connection (currently supported on 8200 only)

Example	NCLOSE(0)
---------	-------------

TCP_OPEN

Purpose To open a TCP/IP connection.

Syntax TCP_OPEN(*N%*, *IP\$*, *RP%*, *LP%* [, *Protocol%*] [, *Delimiter%*])

Remarks **Note that this function must be called before using any socket read/write commands.**

"*N%*" is an integer variable in the range of 0 to 5, indicating the connection number. For FTP, refer to [10.2.1 Command: TCP_OPEN](#).

N%	Meaning
0~3	TCP/IP connection
4	FTP connection (currently supported on 8000, 8200, 8300, 8400, 8700 only)
5	Bluetooth FTP connection (currently supported on 8200 only)

"*IP\$*" is a string variable, indicating the IP address of the remote port. If it is set to "0.0.0.0", the connection will become server mode and the *LP%* must be defined.

"*RP%*" is an integer variable, indicating the port number of the remote port, which is to be connected. It has to be a positive integer in client mode. However, it has to be set to 0 when in server mode.

"*LP%*" is an integer variable, indicating the port number of the local port. It has to be a positive integer in server mode. However, it has to be set to 0 when in client mode.

	Server mode	Client mode	
N%	0 ~ 3	0 ~ 4	5
IP\$	"0,0,0,0"	Required	"0,0,0,0"
RP%	0	Required	0
LP%	Required	0	0

"*Protocol%*" is an integer variable, indicating the networking protocol in use. This parameter is optional and it is set to 0 by default (using TCP/IP protocol). If it is set to 1, the system will use UDP/IP protocol. However, it can only be set to 2 for FTP and Bluetooth FTP.

"*Delimiter%*" is an integer variable, indicating whether to transmit the delimiter or not. This parameter is optional and it is set to 0x0d (Carriage Return) by default. The valid values range from 0 to 255. If it is set to -1, the system will not transmit any delimiter.

Example

```
TCP_OPEN(0, "0.0.0.0", 0, 23)
TCP_OPEN(1, "0.0.0.0", 0, 24)
TCP_OPEN(2, "0.0.0.0", 0, 25, 1)
TCP_OPEN(3, "0.0.0.0", 0, 26, 0, 59)
TCP_OPEN(4, "192.168.6.24", 0, 21, 2, 59)
TCP_OPEN(5, "0.0.0.0", 0, 0, 2, 59)
```

2.3 READ AND WRITE DATA

2.3.1 COMMANDS

NREAD\$

Purpose	To read data from a TCP/IP connection.
Syntax	<code>A\$ = NREAD\$(N%)</code>
Remarks	<p>"A\$" is a string variable to be assigned to the result.</p> <p>"N%" is an integer variable in the range of 0 to 3, indicating the connection number.</p>
Example	<code>A\$ = NREAD\$(0)</code>

NWRITE

Purpose	To write data to a TCP/IP connection.
Syntax	<code>NWRITE(N%, A\$)</code>
Remarks	<p>"N%" is an integer variable in the range of 0 to 3, indicating the connection number.</p> <p>"A\$" is a string variable, representing the string to be sent to the connection.</p>
Example	<code>NWRITE(0, "Hello")</code>

SOCKET_CAN_SEND

Purpose	To check if data can be sent.												
Syntax	<code>A% = SOCKET_CAN_SEND(N%, L%)</code>												
Remarks	<p>"A%" is an integer variable to be assigned to the result.</p> <table border="1"> <thead> <tr> <th>A%</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>Normal – data can be sent</td></tr> <tr> <td>3000</td><td>Invalid connection number</td></tr> <tr> <td>3004</td><td>Connection is closed</td></tr> <tr> <td>3007</td><td>Cannot send data</td></tr> <tr> <td>3012</td><td>Never run START TCPIP</td></tr> </tbody> </table> <p>"N%" is an integer variable in the range of 0 to 3, indicating the connection number.</p> <p>"L%" is an integer variable, indicating the length of data.</p>	A%	Meaning	0	Normal – data can be sent	3000	Invalid connection number	3004	Connection is closed	3007	Cannot send data	3012	Never run START TCPIP
A%	Meaning												
0	Normal – data can be sent												
3000	Invalid connection number												
3004	Connection is closed												
3007	Cannot send data												
3012	Never run START TCPIP												
Example	<code>A% = SOCKET_CAN_SEND(0, 10)</code>												

SOCKET_HAS_DATA

Purpose To check if data is available.

Syntax `A% = SOCKET_HAS_DATA(N%)`

Remarks "A%" is an integer variable to be assigned to the result.

A%	Meaning
0	Normal – data in buffer.
3000	Invalid connection number
3004	Connection is closed
3005	No data
3012	Never run START TCPIP

"N%" is an integer variable in the range of 0 to 3, indicating the connection number.

Example `A% = SOCKET_HAS_DATA(0)`

SOCKET_OPEN

Purpose To check if the remote end of connection is open.

Syntax `A% = SOCKET_OPEN(N%)`

Remarks "A%" is an integer variable to be assigned to the result.

A%	Meaning
0	Normal – connection is open
3000	Invalid connection number
3004	Connection is closed
3012	Never run START TCPIP

"N%" is an integer variable in the range of 0 to 3, indicating the connection number.

Example `ConnectState% = SOCKET_OPEN(0)`

2.4 GET TCP/IP MESSAGE

2.4.1 COMMAND

GET_TCPIP_MESSAGE

Purpose To get the message of TCPIP event trigger.

Syntax A% = GET_TCPIP_MESSAGE

Remarks This command can also be called in normal program to detect the TCP/IP status by polling method. Once it is fetched, the message will be cleared by the system.

When entering TCPIP event trigger, the first thing is to call this routine so that the trigger message will be cleared out.

"A%" is an integer variable to be assigned to the result.

A%	Meaning
4000	Connection #0 overflow
4001	Connection #1 overflow
4002	Connection #2 overflow
4003	Connection #3 overflow
4013	Abnormal break during connection
4014	Networking initialization error
4015	Port initialization error
4020	Connection #0: connected
4021	Connection #1: connected
4022	Connection #2: connected
4023	Connection #3: connected
4040	Connection #0: disconnected
4041	Connection #1: disconnected
4042	Connection #2: disconnected
4043	Connection #3: disconnected
4060	Connection #0: data is coming
4061	Connection #1: data is coming
4062	Connection #2: data is coming
4063	Connection #3: data is coming
4080	IP is ready

Example

```

ON TCPIP GOSUB TCPIP_Trigger
...
TCPIP_Trigger:
MSG% = GET_TCPIP_MESSAGE
...
```


2.5 GET TCP/IP ERROR CODE

2.5.1 COMMAND

TCP_ERR_CODE

Purpose	To check the result after running any command related to TCP/IP (except STOP TCPIP).
Syntax	A% = TCP_ERR_CODE
Remarks	"A%" is an integer variable to be assigned to the result, indicating an error description. If a routine is working normally, the return value will be 0 in general. However, it will return "N%" (the connection number) for TCP_OPEN and "1 ~ 255" (the length of data being read) for NREAD\$.

A%	Meaning
0	Normal
3000	Invalid connection number
3001	Connection is already opened.
3002	Undefined local port in server mode
3003	Undefined remote port in client mode
3004	Connection is closed.
3005	No data received in buffer
3006	Data is too long.
3007	Networking is busy or data is too long.
3008	Data transmission error
3009	Hardware initialization failure
3010	START TCPIP has already been running. Need to run STOP TCPIP.
3011	All connections are unavailable.
3012	Never run START TCPIP
-10	Parameter error
-11	Host is not reachable.
-12	Time out
-13	Hardware failure
-14	Protocol error
-15	No buffer space
-16	Invalid connection block
-17	Invalid pointer argument
-18	Operation would block.
-19	Message is too long.
-20	Protocol unavailable
-30	Unknown remote name

A%	Meaning
-31	DNS protocol error (package class)
-32	DNS protocol error (package type)
-33	Remote name too long (more than 38 characters)

Example

ERR% = TCP_ERR_CODE

WIRELESS NETWORKING

This section describes the commands related to wireless network configuration. These command sets are only applicable to the mobile computers according to their hardware configuration. Refer to [Appendix IV — Examples](#).

- ▶ WLAN stands for IEEE 802.11b/g
- ▶ SPP stands for Serial Port Profile of Bluetooth
- ▶ DUN stands for Dial-Up Networking Profile of Bluetooth for connecting a modem
- ▶ DUN-GPRS stands for Dial-Up Networking Profile of Bluetooth for activating a mobile's GPRS
- ▶ HID stands for Human Interface Device Profile of Bluetooth
- ▶ FTP stands for File Transfer Protocol Profile of Bluetooth
- ▶ GSM stands for Global System for Mobile Communications
- ▶ GPRS stands for General Packet Radio Service
- ▶ UMTS stands for Universal Mobile Telecommunications System
- ▶ HSDPA stands for High Speed Downlink Packet Access

Wireless Product Family						
Mobile Computer	8000	8200	8300	8400	8500	8700
Bluetooth only	8062	8260	8362	8400	8500	8700
WLAN (802.11b/g) only	8071		8370			
Bluetooth + WLAN		8230	8330	8470	8570	8770
Bluetooth + WLAN + WWAN						8790

Note: (1) Refer to the previous section for port mapping of Bluetooth and GSM.
 (2) GSM/GPRS/EDGE or UMTS/HSDPA services are supported on 8700.
 (3) Bluetooth FTP is supported on 8200 only.

IN THIS CHAPTER

3.1 Network Configuration	26
3.2 Initialization & Termination	29
3.3 Network Status.....	32

3.1 NETWORK CONFIGURATION

Before bringing up (initializing) the network, some related parameters must be configured. These parameters are kept by the system during normal operations and power on/off cycles.

3.1.1 IMPLEMENTATION

The parameters can be accessed through System Menu or an application program (via **GET_NET_PARAMETER\$, SET_NET_PARAMETER**). Refer to [Appendix II — Net Parameters by Index](#).

Note: The parameters will be set back to the default values when updating kernel.

3.1.2 COMMANDS

GET_NET_PARAMETER\$

Purpose To get network settings.

Syntax `A$ = GET_NET_PARAMETER$(index%)`

Remarks "A\$" is a string variable to be assigned to the result.

"index%" is an integer variable, indicating a specific configuration item by index number. See [Appendix II — Net Parameters by Index](#).

Error Code	Meaning
0	Normal status: connection is open
3000	Invalid index number
3004	Connection is closed
3012	Never run START TCPIP

Example `NetSetting$ = GET_NET_PARAMETER$(0)`

See Also `SOCKET_IP, TCP_ERR_CODE`

SET_NET_PARAMETER

Purpose To configure network settings.

Syntax SET_NET_PARAMETER(*index%*, *A\$*)

Remarks "*index%*" is an integer variable, indicating a specific configuration item by index number. See [Appendix II — Net Parameters by Index](#).

"*A\$*" is a string variable indicating the network setting to be configured.

A\$	Setup string
(Boolean type)	"Enable" / "Disable"
Authentication	"Open" / "Share"
WEP Key Length	"64 bits" / "128 bits"
System Scale	"Low" / "Medium" / "High" / "Custom-Txrate" / "Custom-RSSI"
Preamble Type	"Short" / "Long" / "Both"
WPA Pass Phrase	8 ~ 63 characters
RoamingTxLimit_11b	"1Mbps" / "2Mbps" / "5.5Mbps" / "11Mbps"
RoamingTxLimit_11g	"1Mbps" / "2Mbps" / "5.5Mbps" / "11Mbps" "6Mbps" / "9Mbps" / "12Mbps" / "18Mbps" "24Mbps" / "36Mbps" / "48Mbps" / "54Mbps"
SCAN_CHANNEL	"bbbbbbbbbbbbbb". b=1 or 0
SCAN_CHANNEL_TIME	"60" / "70" / "80" / "90" / "100" / "110"
ROAMING_RSSI_THRES HOLD	"-50" / "-55" / "-60" / "-65" / "-70" / "-75" / "-80" / "-85" / "-90"
ROAMING_RSSI_DELTA	"0" / "5" / "10" / "15" / "20"
ROAMING_PERIOD	"3" / "4" / "5" / "6" / "7" / "8" / "9" / "10"

For indexes 5 ~ 10, 19, 20, 25, 27, 32, you may simply input an empty string to clear settings.

P_SCAN_CHANNEL, P_SCAN_CHANNEL_TIME, P_ROAM_RSSI_THRHOLD, P_ROAM_RSSI_DELTA, P_ROAM_PERIOD are available only for 8200.

Note that it is not necessary to configure the setting every time.

Example

```

SET_NET_PARAMETER(1, "192.168.1.241") ` set local IP
SET_NET_PARAMETER(11, "Disable")      ` disable DHCP
SET_NET_PARAMETER(22, "Short")         ` set preamble type "Short"
SET_NET_PARAMETER(12, "Share")         ` set authentication "Share
                                         Key"

```

See Also IP_CFG, TCP_ERR_CODE

3.2 INITIALIZATION & TERMINATION

After the networking parameters are properly configured, an application program can call **START TCPIP** to initialize any wireless module (802.11b/g, Bluetooth, or GSM/GPRS) and networking protocol stack.

- ▶ The wireless modules will not be powered until **START TCPIP** is called.
- ▶ When an application program needs to stop using the network, **STOP TCPIP** must be called to shut down the network as well as the modules (so that power can be saved). To enable the network again, **START TCPIP** must be called again.

Note: Any previous network connection and data will be lost after calling STOP TCPIP.

3.2.1 OVERVIEW

8000 Series		
8062	START TCPIP (3)	Enables mobile's GPRS functionality via Bluetooth (DUN)
8071	START TCPIP	Enables 802.11b/g (WLAN)
8200 Series		
8230	START TCPIP	Enables 802.11b/g (WLAN)
	START TCPIP (0)	
	START TCPIP (3)	Enables mobile's GPRS functionality via Bluetooth (DUN)
	START TCPIP (5)	Enables PPP connection via direct RS-232 (to a generic modem)
8260	START TCPIP (3)	Enables mobile's GPRS functionality via Bluetooth (DUN)
	START TCPIP (5)	Enables PPP connection via direct RS-232 (to a generic modem)
8300 Series		
8330	START TCPIP	Enables 802.11b/g (WLAN)
	START TCPIP (0)	
	START TCPIP (3)	Enables mobile's GPRS functionality via Bluetooth (DUN)
	START TCPIP (5)	Enables PPP connection via direct RS-232 (to a generic modem)
8362	START TCPIP (3)	Enables mobile's GPRS functionality via Bluetooth (DUN)
	START TCPIP (5)	Enables PPP connection via direct RS-232 (to a generic modem)
8370	START TCPIP	Enables 802.11b/g (WLAN)
	START TCPIP (5)	Enables PPP connection via direct RS-232 (to a generic modem)

8400 Series		
8400	START TCPIP (3)	Enables mobile's GPRS functionality via Bluetooth (DUN)
	START TCPIP (5)	Enables PPP connection via direct RS-232 (to a generic modem)
8470	START TCPIP	Enables 802.11b/g (WLAN)
	START TCPIP (0)	
	START TCPIP (3)	Enables mobile's GPRS functionality via Bluetooth (DUN)
	START TCPIP (5)	Enables PPP connection via direct RS-232 (to a generic modem)
8500 Series		
8500	START TCPIP (3)	Enables mobile's GPRS functionality via Bluetooth (DUN)
8570	START TCPIP	Enables 802.11b/g (WLAN)
	START TCPIP (0)	
	START TCPIP (3)	Enables mobile's GPRS functionality via Bluetooth (DUN)
8700 Series		
8700	START TCPIP (3)	Enables mobile's GPRS functionality via Bluetooth (DUN)
8770	START TCPIP	Enables 802.11b/g (WLAN)
	START TCPIP (0)	
	START TCPIP (3)	Enables mobile's GPRS functionality via Bluetooth (DUN)
8790	START TCPIP	Enables 802.11b/g (WLAN)
	START TCPIP (0)	
	START TCPIP (2)	Enables 3.5G
	START TCPIP (3)	Enables mobile's GPRS functionality via Bluetooth (DUN)
All Series		
via Modem Cradle	START TCPIP (4)	Enables PPP connection via Modem Cradle
via Ethernet Cradle	START TCPIP (6)	Enables Ethernet connection via Ethernet Cradle

Note: START TCPIP (7) is used to enable GPRS connection via 8400 GPRS Cradle only.

3.2.2 COMMANDS

START TCPIP

Purpose To enable TCP/IP communication.

Syntax START TCPIP
START TCPIP(*N%*)

Remarks This routine is used to perform general initialization. It must be the first network function call, and cannot be called again unless STOP TCPIP has been called.

"*N%*" is an integer variable, indicating which wireless module is to be used.

N%	Meaning
0	802.11b/g (default)
1	Reserved
2	GPRS
3	Mobile's GPRS via Bluetooth (DUN)
4	PPP Connection via Modem Cradle
5	PPP Connection via RS-232
6	Ethernet Connection via Ethernet Cradle
7	GPRS Connection via GPRS Cradle

Example START TCPIP ' this is hardware-dependent

See Also OFF TCPIP, ON TCPIP GOSUB..., STOP TCPIP, TCP_ERR_CODE, TCP_OPEN

STOP TCPIP

Purpose To disable TCP/IP communication.

Syntax STOP TCPIP

Remarks

Example STOP TCPIP

3.3 NETWORK STATUS

Once the network has been initialized, there is some status information can be retrieved from the system. It will be periodically updated by the system. Refer to [Appendix III – Net Status by Index](#).

Note: (1) User program must explicitly call GET_NET_STATUS to get the latest status.
(2) If GET_NET_STATUS(7) returns -1, it means an abnormal break occurs during PPP, DUN-GPRS, or GPRS connection. Such disconnection may be caused by the mobile computer being out of range, improperly turned off, etc.

3.3.1 COMMAND

GET_NET_STATUS

Purpose	To get network status.
Syntax	<code>A% = GET_NET_STATUS(index%)</code>
Remarks	<p>"A%" is an integer variable to be assigned to the result.</p> <p>"index%" is an integer variable indicating a specific configuration item by index number.</p> <p>Note that it is necessary to define the DNS server IP before running this command.</p> <p>Please refer to the table below.</p>
Example	<code>nQuality = GET_NET_STATUS(2)</code> ` check communication quality

IEEE 802.11B/G/N

IEEE 802.11b/g/n is an industrial standard for Wireless Local Area Networking (WLAN), which enables wireless communications over a long distance. The speed of connection between two wireless devices will vary with range and signal quality.

To maintain a reliable connection, the data rate of the 802.11b/g/n system will automatically fall back as range increases or signal quality decreases.

802.11 Specification	
Frequency Range:	2.4 GHz
Data Rate:	802.11b - 1, 2, 5.5, 11 Mbps 802.11g - 6, 9, 12, 18, 24, 36, 48, 54 Mbps 802.11n - 6.5, 13, 19.5, 26, 39, 52, 58.5, 65 Mbps
Connected Devices:	1 for ad-hoc mode (No AP) Multiple for infrastructure mode (AP required)
Protocol:	IP/TCP/UDP
Max. Output Power:	50 mW (802.11b)
Spread Spectrum:	DSSS/OFDM
Modulation:	802.11b - DBPSK, DQPSK, CCK 802.11g - BPSK, QPSK, 16-QAM, 64-QAM 802.11n - BPSK, QPSK, 16-QAM, 64-QAM
Standard:	IEEE 802.11b/g/n, interoperable with Wi-Fi devices

Note: All specifications are subject to change without prior notice. IEEE 802.11n is only for 8231.

IN THIS CHAPTER

4.1 Obsolete Command	34
4.2 Wi-Fi Profile	34
4.3 Scanning for Wi-Fi Hotspots	38

4.1 OBSOLETE COMMAND

Note: For the stability and compatibility concern, it is recommended to use **GET_NET_STATUS**.

GET_WLAN_STATUS

Purpose	To get network status.
Syntax	<code>A% = GET_WLAN_STATUS(index%)</code>
Remarks	This command is to be replaced by GET_NET_STATUS.
Example	<code>nQuality = GET_WLAN_STATUS(2)</code> ` check communication quality

4.2 WI-FI PROFILE

For a prompt connection via Wi-Fi device, you can configure pre-settings into a profile. Base on variety configurations including AP, Authentication, Security, Preamble Type and so on. Here supports you up to four vacant profiles used for saving the pre-settings.

4.2.1 COMMANDS

GET_NET_PARAMETER\$

8200, 8700

Purpose	To get Profile configurations.				
Syntax	<code>A\$ = GET_NET_PARAMETER\$(index%)</code>				
Remarks	<p>"A\$" is a string variable to be assigned to the result.</p> <p>"index%" is an integer variable, indicating a specific configuration item by index number. See Appendix II — Net Parameters by Index.</p> <table border="1"> <thead> <tr> <th>Index</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>-84~-87</td><td>Read PROFILEX</td></tr> </tbody> </table>	Index	Meaning	-84~-87	Read PROFILEX
Index	Meaning				
-84~-87	Read PROFILEX				
Example	<code>A\$=GET_NET_PARAMETER\$(-84)</code> ` Read Profile1				
See Also	TCP_ERR_CODE				

SET_NET_PARAMETER**8200, 8700**

Purpose To set Profile configurations.

Syntax SET_NET_PARAMETER(*index%*, A\$)

Remarks "*index%*" is an integer variable, indicating a specific configuration item by index number. See [Appendix II — Net Parameters by Index](#).
 "A\$" is an empty or string variable indicating the configured network settings to be saved.

Index	A\$	Setup String
84~87	" "	If A\$ is a empty string, save active setting to PROFILEx
	Setting String	Save A\$ to Profilex
88~91	" "	If A\$ is a empty string, load PROFILEx to active setting
	Setting String	Load A\$ to active setting

Example A\$="55,1,1,1,1,1,1234567890,9988776655,4433221122,5555566666"

SET_NET_PARAMETER(84,A\$) ` save A\$ to PROFILE1

SET_NET_PARAMETER(88," ") ` Apply PROFILE1

See Also TCP_ERR_CODE

Setting String Forma(A\$)

A\$ = Basic Items + Security Items

Basic Items:

<i>SSID (Length: 32)</i>	<i>BSSTYPE (Length: 1)</i>	<i>Security (Length: 1)</i>
	0: Ad-Hoc 1: Infrastructure	0: None 1: WEP OpenSystem 2: WEP ShareKey 3: WPA-PSK 4: WPA2-PSK 5: EAP

Security Items 1 or 2 – WEP:

<i>WEPLen (Length: 1)</i>	<i>Defaultkey (Length: 1)</i>	<i>Key1 (Length: 14)</i>	<i>Key2 (Length: 14)</i>	<i>Key3 (Length: 14)</i>	<i>Key4 (Length: 14)</i>
0: 64bit 1: 128bit	0: key1 1: key2 2: key3 3: key4	Length: 5 or 13	Length: 5 or 13	Length: 5 or 13	Length: 5 or 13

Note: Input the WEP security key with Ascii and the length of WEP key must be specified to 5 or 13 characters.

Security Items 3 or 4 – WPA:

<i>WPA Passphrase (Length: 64)</i>

Security Items 5 – EAP:

<i>EAP ID (Length: 33)</i>	<i>EAP Password (Length: 33)</i>

Example 1:

SSID: 55

BSSTYPE: 1

Security: 1 (WEP OpenSystem)

WEPLen: 1 (128bit)

Defaultkey: 1 (Key2)

Key1: 12345

Key2: 99887

Key3: 44332

Key4: 55555

A\$= "55,1,1,1,1,1,12345,99887,44332,55555"

A\$= "55,1,1,1,3,,,,,55555" (only configure key 4)

Example 2:

SSID: 66

BSSTYPE: 1

Security: 3 (WPA-PSK)

WPA Passphrase: 1234567890123456789012345678901234567890

A\$= "66,1,3,1234567890123456789012345678901234567890"

Example 3:

SSID: aAbBcCdD

BSSTYPE: 1

Security: 5 (EAP)

EAP ID: 111222333

EAP Password: 777888999

A\$= "aAbBcCdD,1,5,111222333,777888999"

4.3 SCANNING FOR WI-FI HOTSPOTS

WIFI_SCAN	8200, 8400, 8700
------------------	-------------------------

Purpose To scan Wi-Fi hotspots.

Syntax $A\$ = \text{WIFI_SCAN}(N1\%, N2\%)$

Remarks "N1%" is an integer variable used as parameter1.
"N2%" is an integer variable used as parameter2.
"A\$" is a string variable to hold the result.

N1	Meaning	N2	Meaning	A\$
1	Scan hotspots	1~8	Maximum number of hotspots to search	Number of found hotspots
2	Read information about the found hotspot		Index of the hotspot	Information about the hotspot

Structure of the information string

Offset of A\$	Meaning
1~2	Length of SSID
3~34	SSID, max 32bytes
35~46	BSSID
47~48	RSSI
49~50	Channel
51	Band Type. "1" - 802.11b/g, "2" - 802.11b
52	BSS Type. "1" - Ad-hoc, "2" - Infrastructure
53	Security. "0" - none, "1" - WEP, "2" - WPA, "4" - WPA2, "6" - WPA/WPA2

Example

```

A$=WIFI_SCAN(1,5)           \ Scan 5 hotspots
B%=ASC(A$)
IF B%>=2 THEN                \ IF find more than 2 hotspots
    W1$=WIFI_SCAN(2,1)       \ Read hotspot1
    PRINT W1$
    W2$=WIFI_SCAN(2,2)       \ Read hotspot2
    PRINT W2$
END IF

```

BLUETOOTH

Refer to [Appendix IV — Examples](#).

Bluetooth Specification	
<i>Frequency Range:</i>	2.4 GHz
<i>Profiles:</i>	SPP, DUN, HID, FTP
<i>Spread Spectrum:</i>	FHSS
<i>Modulation:</i>	GFSK
<i>Standard:</i>	Bluetooth version 2.0 + EDR

Note: All specifications are subject to change without prior notice.

IN THIS CHAPTER

5.1 Bluetooth Profiles Supported	40
5.2 Frequent Device List.....	41
5.3 Inquiry	43
5.4 Pairing.....	44

5.1 BLUETOOTH PROFILES SUPPORTED

Serial Port Profile (SPP)
For ad-hoc networking, without going through any access point.
Dial-Up Networking Profile (DUN)
For a mobile computer to make use of a Bluetooth modem or mobile phone as a wireless modem. Also, it can be used to activate the GPRS functionality on a mobile phone.
Human Interface Device Profile (HID)
For a mobile computer to work as an input device, such as a keyboard for a host computer.
File Transfer Protocol Profile (FTP)
For a mobile computer to connect to a file server for file transfer.

Note: Bluetooth FTP is supported on 8200 only.

5.2 FREQUENT DEVICE LIST

Through the pairing procedure, the mobile computer is allowed to keep record of the latest connected device(s) for different Bluetooth services, regardless of authentication enabled or not. Such record is referred to as "Frequent Device List".

Service Type		In Frequent Device List
Serial Port	SPP	Only 1 device is listed for quick connection.
Dial-up Networking	DUN	Only 1 device is listed for quick connection.
Human Interface Device	HID	Only 1 device is listed for quick connection.
File Transfer	FTP	Only 1 device is listed for quick connection.

Get Frequent Device List

The length of Frequent Device List by calling **GET_NET_PARAMETER\$** is 83 characters:

LIST\$ = GET_NET_PARAMETER\$(-40)

Length	Properties	Char												
1	Service Type	1	13 2											
2 ~ 13	MAC ID	12	"0"	"0"	"D"	"0"	"1"	"7"	"3"	"0"	"1"	"2"	"3"	"4"
14 ~ 33	Device Name	20	"M"	"Y"	" "	"N"	"A"	"M"	"E"	0	0
34 ~ 50	PIN Code	17	"1"	"2"	"3"	"4"	0	0
51 ~ 83	Link Key	33	"1"	"2"	"4"	"F"	"5"	"3"	0

- ▶ The first character of Frequent Device List is the service type that the device is engaged. Currently, there are four types that have been defined:

Service Type		In Frequent Device List
3	SPP	Only 1 device is recorded.
4	DUN	Only 1 device is recorded.
5	HID	Only 1 device is recorded.
6	Reserved	
7	FTP	Only 1 device is recorded.

Note: If bit 7 = 1, it means that this device is currently connected.

- ▶ After the service type, from 2nd to the 13th character stands for the string of MAC ID.
- ▶ The next property after MAC ID is Device Name, which consists of up to 20 characters and ends with a delimiter code "\r".
- ▶ The next property after Device Name is PIN code, which consists of up to 17 characters and ends with a delimiter code "\r".
- ▶ The last property of Frequent Device List is Link Key, which is normally generated when the pairing procedure is completed. This unique Link Key is applied to the specific device connection only. Once the connection is renewed with a different device, a new Link Key will be generated.

Note: Make sure to put “\r” as a delimiter for Device Name, PIN Code, and Link Key.

Sample code:

```
FREQ_DEV$ = ""
CLS
FOR K% = 1 TO 8
    FDL$ = ""
    FDL$ = GET_NET_PARAMETER$(-39-K%)
    IF MID$(FDL$, 1, 1) <> CHR$(0) THEN
        DEV$ = MID$(FDL$, 14, 20)
        MAC_ID$ = MID$(FDL$, 2, 12)
        MACHINE$ = MID$(FDL$, 1, 1)
        FREQ_DEV$ = FREQ_DEV$+DEV$
        FREQ_MAC$ = FREQ_MAC$+MAC_ID$
        FREQ_MC$ = FREQ_MC$+MACHINE$
    END IF
NEXT K%
I% = MENU(FREQ_DEV$)
```

Set Frequent Device List

To enable quick connection to a specific device without going through the inquiry and pairing procedure, a user-definable Frequent Device List can be set up by calling **SET_NET_PARAMETER**.

- ▶ If there is an existing Frequent Device List generated from the inquiry and pairing procedure, it then may be partially or overall updated by this, and vice versa.
- ▶ There are five fields: Service Type, MAC ID, Device Name, PIN Code, and Link Key.
- ▶ If authentication is disabled, you only need to specify the first three fields. Otherwise, the PIN code field needs to be specified for generating Link Key.

Sample code (1):

```
' setting up a DUN Frequent Device List without authentication
' by calling SET_NET_PARAMETER:
...
FDL$ = CHR$(4+128) + "00d017401234" + "TestDev." + CHR$(13)
SET_NET_PARAMETER(40, FDL$)
...
```

sample code (2):

```
' setting up a SPP Frequent Device List with authentication
' (needs PIN code) by calling SET_NET_PARAMETER:
...
FDL$ = CHR$(3+128) + "00d017401234" + "TestDev." + CHR$(13) + "1234"+CHR$(13)
SET_NET_PARAMETER(40, FDL$)
```

5.3 INQUIRY

To complete the pairing procedure, it consists of two steps: (1) to discover the Bluetooth devices in range, and (2) to page one of them that provides a particular service. These are handled by **BT_INQUIRY\$** and **BT_PAIRING** respectively.

- ▶ Once the pairing procedure is completed and the list is generated, next time the mobile computer will automatically connect to the listed device(s) without going through the pairing procedure.

5.3.1 COMMAND

BT_INQUIRY\$

Purpose	To discover any available Bluetooth devices in range.
Syntax	A\$ = BT_INQUIRY\$
Remarks	<p>It takes about 20 seconds to get the Bluetooth device information of whomever in range. The string contains address (12 bytes) and name (20 bytes) of the devices.</p> <p>Note that there might be many devices concatenated together, each occupying 32 bytes. Information regarding the Bluetooth devices in range will be put in the format of MENU() string as shown below:</p>

Device address 12 characters												Device name max. = 19 characters + 0x0d																											
0	0	D	0	1	7	4	0	0	1	2	3	C	I	P	H	E	R	L	A	B	\r	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗						
0	0	D	0	1	7	4	0	0	2	1	4	8	3	0	0	-	9	3	1	8	8	7	\r	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗					
0	0	A	9	3	6	2	8	4	5	3	1	P	R	I	N	T	E	R	\r	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗						
0	0	B	0	1	3	4	A	5	2	B	4	W	E	N	'	P	C	\r	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗						
⊗ = NULL																																							

⊗ = NULL

Example ...
 MENU_STR\$ = BT_INQUIRY\$
 I% = MENU(MENU_STR\$)
 ...

5.4 PAIRING

According to the search results for nearby Bluetooth devices, the application can then try to pair with any of the remote devices by calling **BT_PAIRING**.

5.4.1 COMMAND

BT_PAIRING

Purpose To check if the discovered device can provide a specific type of service, and (if required), the PIN code for authentication is matching.

Syntax `A% = BT_PAIRING(addr$, type%)`

Remarks "A%" is an integer variable to be assigned to the result.

A%	Meaning
1	Pairing successfully
0	Service unavailable or wrong PIN code

"addr\$" is a string variable, indicating the address of the Bluetooth device.

"type%" is an integer variable, indicating a specific type of service.

TYPE%	Meaning
1	Reserved
3	SPP
4	DUN
6	Reserved
7	FTP

It will try to pair with any Bluetooth device that has the specific type of service. If authentication is enabled, then correct PIN code will be required for setting up the Link Key. Once the pairing procedure is completed, the MAC ID of the remote device will be recorded in the "Frequent Device List" for quick connection in the future.

Example

```
...  
MENU_STR$ = BT_INQUIRY$  
I% = MENU(MENU_STR$)  
DEVICE$ = MID$(MENU_STR$, 1+32*(I%-1), 32)  
R% = BT_PAIRING(DEVICE$, 3)  
...
```

GSM/GPRS

This section describes the BASIC commands related to GSM/GPRS. Currently, these commands are for the use of 8700 Series.

Data services of GSM, including SMS (Short Message Service) and data call, are provided for receiving and sending data. They are performed via a virtual COM port, namely, *COM3*. The communication type, *GSM_SMS* and *GSM_Modem*, which are for SMS and data call respectively, should be assigned by calling **SET_COM_TYPE** before use. The *GSM_SMS* supports uncompressed PDU (Protocol Description Unit) message mode. It can handle both 7-bit default alphabet and 8-bit data. In addition, concatenated messages are also supported. Refer to [Appendix IV — Examples](#).

Note: GSM/GPRS/EDGE or UMTS/HSDPA services are supported on 8700.

IN THIS CHAPTER

6.1 Data Format	46
6.2 Security	49
6.3 PIN Code	51

6.1 DATA FORMAT

READ_COM\$ data format

For SMS service, the data format for single messages and concatenated messages is different. The short messages will be removed from the SIM card after being read out. If it is necessary to save the received data, data storage structure like a DAT or DBF file is recommended.

Message Type	Single Message	Concatenated Message
Using 7-bit default alphabet	total length \leq 160 characters	total length $>$ 160 characters
Using 8-bit	total length \leq 140 octets	total length $>$ 140 octets
Using 16-bit	total length \leq 70 characters	total length $>$ 70 characters

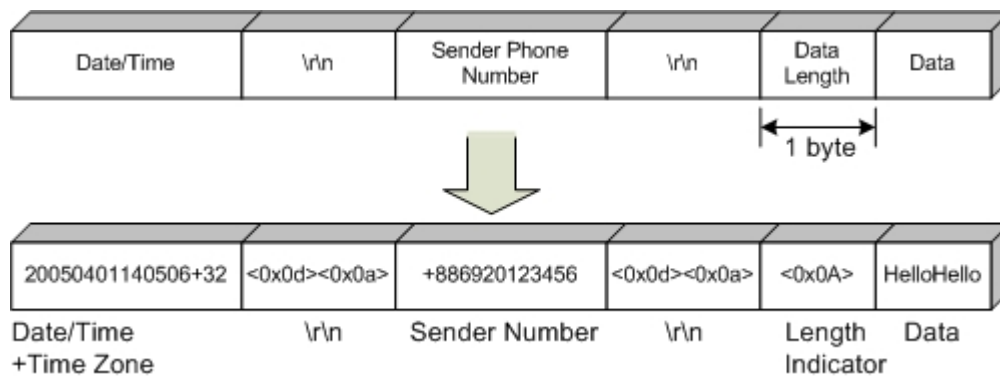
► Single Message:

The diagram below shows the data format for a single message received by calling **READ_COM\$**. The data length is the number of octets of data.

Example:

20050401140506+32<0x0d><0x0a>+886920123456<0x0d><0x0a><0x0A>

HelloHello



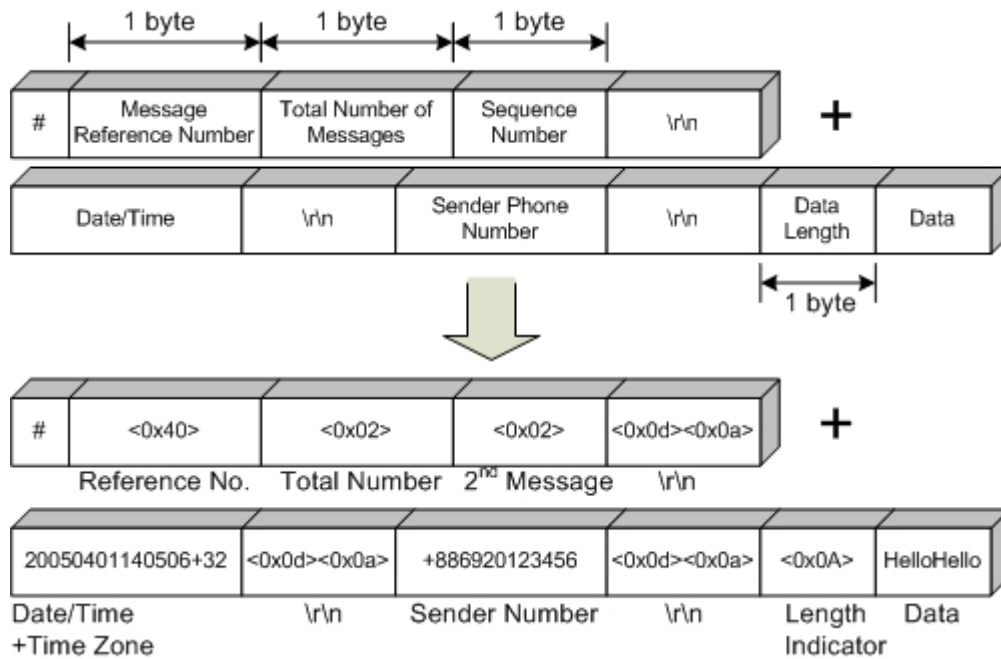
► Concatenated Message

The whole data will be separated into several sections.

The diagram below shows the data format for a concatenated message received by calling **READ_COM\$**. The data length is the number of octets of data.

Example:

```
#<0x40><0x02><0x02><0x0d><0x0a>20050401140506+32<0x0d><0x0a>
+886920123456<0x0d><0x0a><0x0A>HelloHello
```



WRITE_COM data format

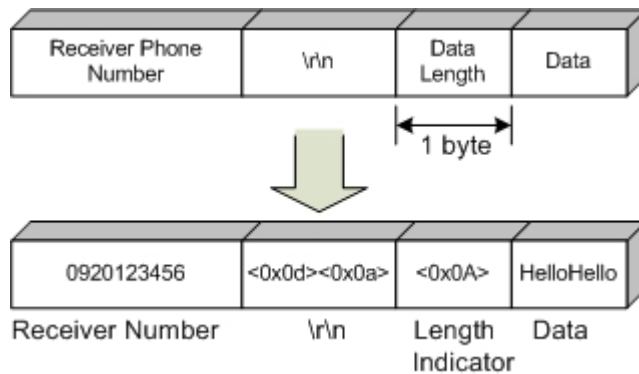
For sending a message, the maximum length is limited to 255 characters.

- ▶ For long messages (see Message Type - Concatenated Message above), data will be sent successfully by using **WRITE_COM**, and then each message will be separated into sections intentionally.

The sending data buffer will not be overwritten until **GET_NET_STATUS(13)** returns 1 to indicate the transmission is completed.

The data format for sending a message is as shown below.

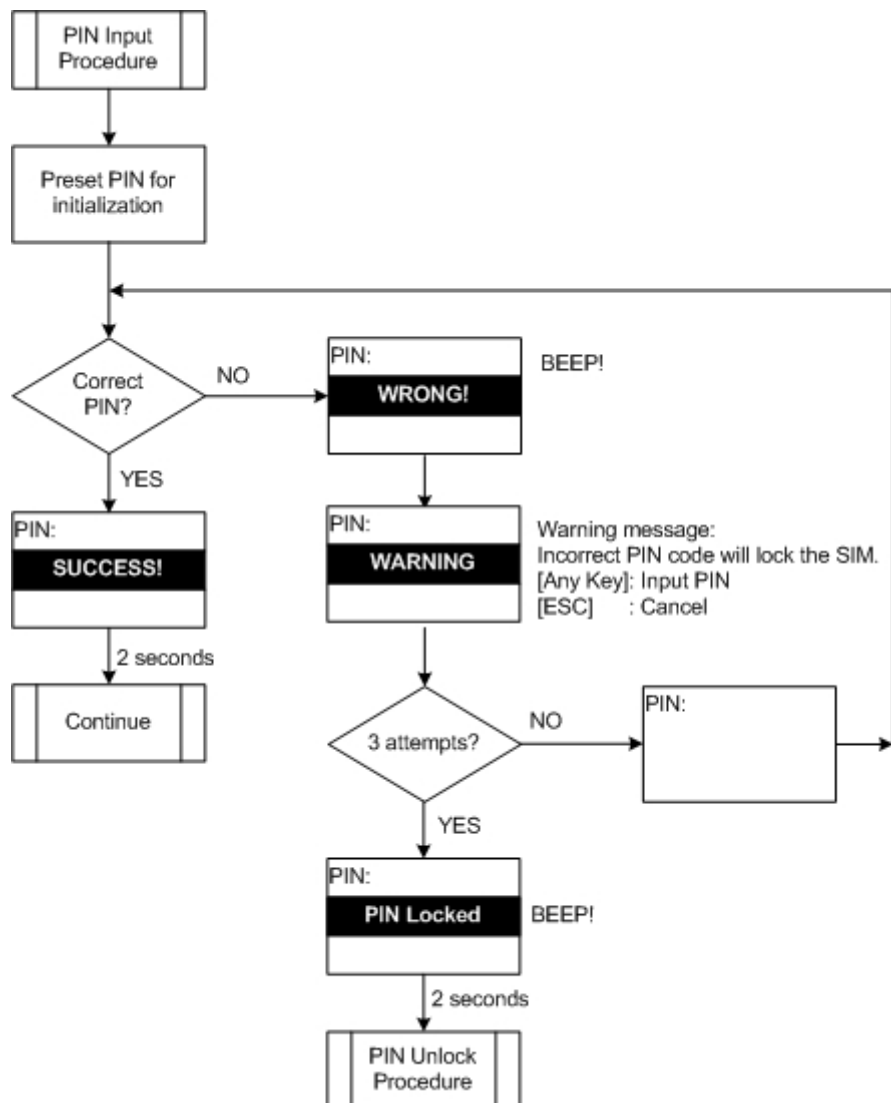
Example: 0920123456<0x0d><0x0a><0x0A>HelloHello



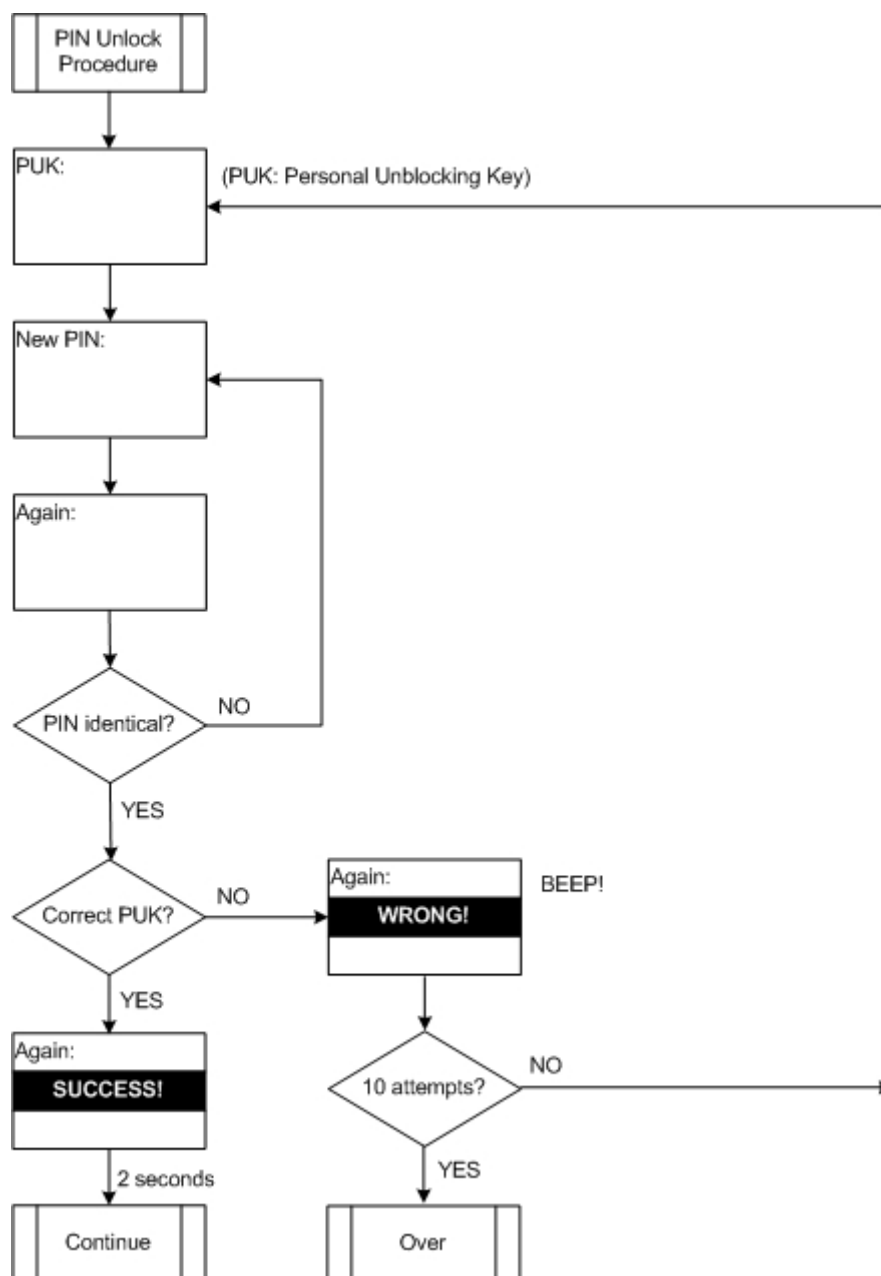
6.2 SECURITY

PIN (Personal Identity Number) is a 4-8 digit access code which can be used to secure your SIM card from use. If the wrong PIN is entered in more than three times, the SIM card will be locked. PUK (Personal Unblocking Key) is an 8-digit code used to unlock the PIN code if your SIM card is blocked. Contact your service provider for PUK. If the wrong PUK is entered ten times in a row, the device will become permanently blocked and unrecoverable, requiring a new SIM card.

6.2.1 PIN PROCEDURE



6.2.2 PUK PROCEDURE



6.3 PIN CODE

6.3.1 COMMANDS

GSM_CHANGE_PIN

8790

Purpose To change the PIN code.

Syntax `A% = GSM_CHANGE_PIN(old$, new%)`

Remarks "A%" is an integer variable assigned to the result.

A%	Meaning
1	The new PIN code has been accepted.
0	The old PIN code is incorrect.
-1	The GSM/GPRS module is running.
-2	Hardware error occurs.
-5	The request times out.

Note that this command cannot be executed while the GSM/GPRS module is running. The old PIN string must be the password of facility. In this case, the new PIN code can be adopted and the remaining attempt counter of PIN will be reset to 3. If the old code is wrong, not only the password cannot be changed successfully, but also the counter will be decremented by 1.

Example `GSM_CHANGE_PIN(PIN1$, PIN2%)` ' to change PIN code from PIN1 to PIN2.

GSM_CHECK_PIN

8790

Purpose To verify the PIN code.

Syntax `A% = GSM_CHECK_PIN(pin%)`

Remarks "A%" is an integer variable assigned to the result.

A%	Meaning
2	No PIN code is required.
1	The new PIN code has been accepted.
0	The old PIN code is incorrect.
-1	The GSM/GPRS module is running.
-2	Hardware error occurs.
-6	The PUK procedure is required.

Note that this command cannot be executed while the GSM/GPRS module is running. If the input code is the correct PIN code, the remaining attempt counter of PIN is reset to 3. If the old code is wrong, the counter will be decremented by 1.

Example `GSM_CHECK_PIN(PIN1%)` ' to verify whether the PIN code is PIN1 or not

GSM_SET_PINLOCK**8790**

Purpose To decide whether to lock the SIM card or not.

Syntax $A\% = \text{GSM_SET_PINLOCK}(pin\$, mode\%)$

Remarks "A%" is an integer variable assigned to the result.

A%	Meaning
1	The new PIN code has been accepted.
0	The old PIN code is incorrect.
-1	The GSM/GPRS module is running.
-2	Hardware error occurs.
-3	The PIN code has already been locked.
-4	The PIN code has already been unlocked.
-5	The request times out.
MODE%	Meaning
0	Unlock the SIM card
1	Lock the SIM card

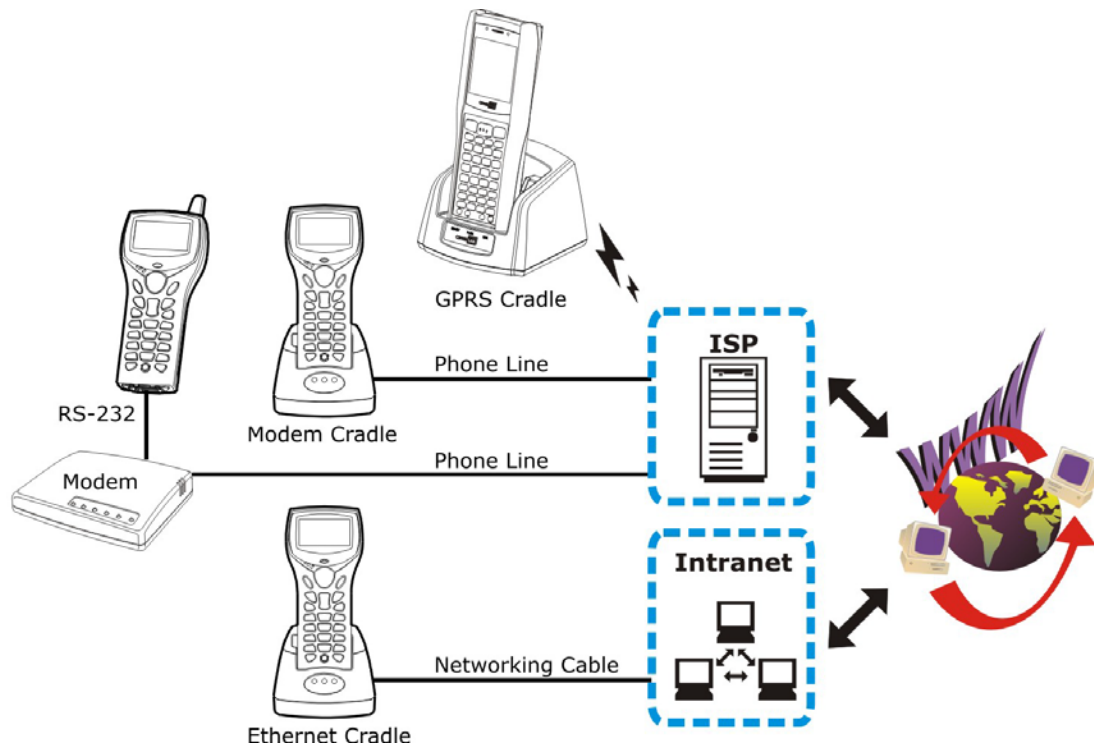
Note that this command cannot be executed while the GSM/GPRS module is running. If the PIN code lock is already enabled (or disabled), one further enabling (or disabling) execution does not take effect.

For an unlocking process, the correct PIN code is required. Otherwise, the execution will fail and the remaining attempt counter of PIN will be decremented by 1. For a locking process, the old PIN code used before unlocking is required. Otherwise, the execution will fail and the counter will be decremented by 1.

Example `GSM_SET_PINLOCK(PIN1$, 1)` ' to lock the SIM card, using PIN code "PIN1\$"

MODEM, ETHERNET & GPRS CONNECTION

Below are available libraries that support (1) PPP connection over serial links, (2) Ethernet connection (Transparent mode), and (3) GPRS connection (Transparent mode). Refer to [Appendix IV — Examples](#).



Note: GPRS (Transparent mode) is currently supported on 8400, with use of GPRS Cradle. Cradle firmware must be version 1.01 or later.

IN THIS CHAPTER

7.1 PPP via Modem Cradle/RS-232.....	54
7.2 Ethernet via Cradle	55
7.3 GPRS via Cradle.....	56

7.1 PPP VIA MODEM CRADLE/RS-232

PPP, short for Point-to-Point Protocol, is a method of connecting the mobile computer to the Internet over serial links. It sends TCP/IP packets to a server that connects to the Internet.

PPP Connection via Modem Cradle

It is supported when making use of the proprietary modem cradle. For baud rate setting, any value other than 57600 bps (default) must be configured through the DIP switch of the IR control board.

Note: For 8000/8300 Series, the version of IR control board on the modem cradle must be greater than SV3.01.

PPP Connection via RS-232

It is supported on 8200/8300/8400/8700 only when being connected to a generic modem (direct RS-232).

7.1.1 CONFIG SETTINGS

Follow the same programming flow of [WLAN Example \(802.11b/g\)](#). Before calling **START TCPIP(4)** or **START TCPIP(5)**, the following parameters of PPP must be specified.

Index		Configuration Item	Default	Description
-70	70	P_PPP_DIALUPPHONE [20]	---	Phone number of ISP
-71	71	P_PPP_LOGINNAME [41]	---	Login user name of ISP
-72	72	P_PPP_LOGINPASSWORD [20]	---	Login password of ISP
-73	73	P_PPP_BAUDRATE (cf. SET_COM)	1	Baud rate matching modem cradle or modem

Note: For baud rate values of IR or RS-232, see the baud rate parameter in SET_COM.

7.2 ETHERNET VIA CRADLE

It is supported when making use of the proprietary Ethernet cradle. First, configure the Ethernet cradle to work in "Transparent" mode. Then, follow the same programming flow of [WLAN Example \(802.11b/g\)](#) using **START TCPIP(6)**.

Refer to the Ethernet Cradle manual for more information on the working modes.

7.3 GPRS VIA CRADLE

It is supported when making use of the 8400 GPRS Cradle. Use AT commands to configure PIN code and GPRS AP name. Then, follow the same programming flow of [WLAN Example \(802.11b/g\)](#) using **START TCPIP(7)**. It fails to initialize a connection in the following conditions: (1) PIN code and GPRS AP name are not configured correctly via AT commands, and (2) CHAP settings are not configured correctly on 8400.

Refer to the 8400 GPRS Cradle manual for more information on the working modes.

USB CONNECTION

Applications are to read and/or write data via a virtual COM port, namely, *COM5*. The communication types should be assigned by calling **SET_COM_TYPE** before use. Before calling **OPEN_COM(5)**, the following parameters of USB must be specified.

Index		Configuration Item	Default	Description
-80	80	P_USB_VCOM_BY_SN	Disable	USB Virtual COM port varies by serial number

Refer to [Appendix IV — Examples](#).

IN THIS CHAPTER

8.1 Overview	58
--------------------	----

8.1 OVERVIEW

8.1.1 USB HID

For 8200/8400/8700 Series, it can be set to work as an input device, such as a keyboard for a host computer.

8.1.2 USB VIRTUAL COM

USB Virtual COM

For 8200/8400/8700 Series, when USB Virtual COM is in use, it is set to use one Virtual COM port for all (USB_VCOM_FIXED) whenever connecting more than one mobile computer to PC via USB. This setting requires you to connect one mobile computer at a time, and will facilitate configuring a great amount of 8200/8400/8700 mobile computers via the same Virtual COM port (for administrators' or factory use). If necessary, you can have it set to use variable Virtual COM port (USB_VCOM_BY_SN), which will vary by the serial number of each different mobile computer.

USB Virtual COM_CDC

For 8200/8700 Series, when USB Virtual COM_CDC is in use, it is set to use one Virtual COM_CDC port for all (USB_VCOM_FIXED) whenever connecting more than one mobile computer to PC via USB. This setting requires you to connect one mobile computer at a time, and will facilitate configuring a great amount of mobile computers via the same Virtual COM_CDC port (for administrators' or factory use). If necessary, you can have it set to use variable Virtual COM_CDC port (USB_VCOM_BY_SN), which will vary by the serial number of each different mobile computer.

8.1.3 USB MASS STORAGE DEVICE

When 8200/8400/8700 Series is equipped with SD card and connected to your computer via the USB cable, it can be treated as a removable disk as long as it is configured properly through programming or System Menu.

GPS FUNCTIONALITY

8700 supports GPS functionality as long as the GPS module is present. Call **OPEN_COM(6)**, **SYSTEM_INFORMATION\$(index)**, and then **CLOSE_COM(6)**.

The information on GPS speed, latitude, longitude and altitude is not confirmed until the return value of GPS status becomes 1.

Index	SYSTEM_INFORMATION\$(index)	Meaning
21	GPS Status	GPS status
22	GPS Speed	Your speed when heading toward a target (relative speed, km/h)
23	GPS Latitude	Your location on earth by latitude coordinates (N for North, S for South): <ul style="list-style-type: none">▶ ddmm.mmmmN or ddmm.mmmmS▶ For example, 1211.1111N means 12° 11' 6.67" North.
24	GPS Longitude	Your location on earth by longitude coordinates (E for East, W for West): <ul style="list-style-type: none">▶ dddmm.mmmmE or dddmm.mmmmW▶ For example, 2326.2141E means 23° 26' 12.85" East.
25	GPS SNR	Signal to Noise ratio, average (dB)
26	GPS Satellite Number	Number of satellites found
27	GPS Altitude	Your location on earth by altitude (meters)

FTP FUNCTIONALITY

File Transfer Protocol (FTP), which runs over Transmission Control Protocol (TCP), is used to transfer files over any network that supports TCP/IP regardless of operating systems. The FTP functions provided here are for the 8000/8200/8300/8400/8700 Mobile Computers to log in to any FTP server and log out over network. During a valid session, the mobile computer can issue commands to the server to perform a specific task, such as create, change or remove directories on the server, delete, upload or download files, etc.

Below lists the BASIC commands used to start an FTP session.

- ▶ Call **FTP_ROUTINES\$()** to get information on the working directory, change directory, or transfer files.
- ▶ Call **SET_NET_PARAMETER()** to configure user name and password.
- ▶ Call **TCP_OPEN()** to open a connection and log on to the host.
- ▶ Call **NCLOSE()** to close the connection.

Note: Only one connection is allowed at a time.

Compiler and Runtime

Any FTP application written in BASIC language requires a certain version of the runtime program specific to the mobile computer that is capable of wireless connectivity:

Mobile Computer	8071	8230	8330, 8370	8470
BASIC Runtime	Version 4.15 or later	Version 1.00 or later	Version 4.08 or later	Version 1.10 or later

Note: BASIC Compiler must be version 4.17 or later.

IN THIS CHAPTER

10.1 Configure Settings.....	62
10.2 Connect and Disconnect	63
10.3 Do FTP Task	68
10.4 Download Program Updates.....	71
10.5 File Handling.....	77
10.6 SD Card Access.....	81

10.1 CONFIGURE SETTINGS

10.1.1 NET PARAMETERS BY INDEX

Index		Configuration Item	Default Setup String	FTP
GET_NET_PARAMETER\$	SET_NET_PARAMETER			
-81	81	FTP_USERNAME [65]	---	✓
-82	82	FTP_PASSWORD [65]	---	✓

10.1.2 COMMAND: GET_NET_PARAMETER

GET_NET_PARAMETER\$ **8000, 8200, 8300, 8400, 8700**

Purpose To get network settings.

Syntax `A$ = GET_NET_PARAMETER$(index%)`

Remarks "A\$" is a string variable to be assigned to the result.

"index%" is an integer variable, indicating a specific configuration item by index number. See [10.1.1 Net Parameters by Index](#).

Error Code	Meaning
0	Normal status: connection is open
3000	Invalid index number
3004	Connection is closed
3012	Never run START TCPIP

Example `UserName$ = GET_NET_PARAMETER$(-81)`

`Password$ = GET_NET_PARAMETER$(-82)`

10.1.3 COMMAND: SET_NET_PARAMETER

SET_NET_PARAMETER **8000, 8200, 8300, 8400, 8700**

Purpose To configure network settings.

Syntax `SET_NET_PARAMETER(index%, A$)`

Remarks "index%" is an integer variable, indicating a specific configuration item by index number. See [10.1.1 Net Parameters by Index](#).

"A\$" is a string variable indicating the network setting to be configured.

Note that it is not necessary to configure the setting every time.

Example `SET_NET_PARAMETER(81, "Test")` ' set login user name

`SET_NET_PARAMETER(82, "1234")` ' set password

10.2 CONNECT AND DISCONNECT

Use **TCP_OPEN(4, ...)** to open a connection and log on to the host over network. Refer to [10.1 Configure Settings](#) for configuring username and password.

10.2.1 COMMAND: TCP_OPEN

TCP_OPEN	8000, 8200, 8300, 8400, 8700
----------	------------------------------

Purpose To open an FTP connection via 802.11b/g. For 8200, it also supports connecting via Ethernet Cradle or Bluetooth. See [Bluetooth FTP example](#).

Syntax TCP_OPEN(4, IP\$, RP%, LP% [, Protocol%] [, Delimiter%])

Remarks **Note that this function must be called before using any socket read/write commands.**

"N%" is "4" for FTP connection and "5" for Bluetooth FTP connection.

N%	Meaning
4	FTP connection via 802.11b/g or Ethernet Cradle (Ethernet Cradle currently supported for 8200 only)
5	Bluetooth FTP connection (currently supported on 8200 only)

"IP\$" is a string variable, indicating the IP address of the remote port. If it is set to "0.0.0.0", the connection will become server mode and the LP% must be defined.

"RP%" is an integer variable, indicating the port number of the remote port, which is to be connected. It has to be a positive integer in client mode. However, it has to be set to 0 when in server mode.

"LP%" is an integer variable, indicating the port number of the local port. It has to be a positive integer in server mode. However, it has to be set to 0 when in client mode.

	Server mode	Client mode	
N%	0 ~ 3	0 ~ 4	5
IP\$	"0,0,0,0"	Required	"0,0,0,0"
RP%	0	Required	0
LP%	Required	0	0

"Protocol%" is an integer variable, indicating the networking protocol in use. This parameter is optional and it is set to 0 by default (using TCP/IP protocol). If it is set to 1, the system will use UDP/IP protocol. However, it can only be set to 2 for FTP and Bluetooth FTP.

"Delimiter%" is an integer variable, indicating whether to transmit the delimiter or not. This parameter is optional and it is set to 0x0d (Carriage Return) by default. The valid values range from 0 to 255. If it is set to -1, the system will not transmit any delimiter.

Example

```
START TCPIP          'select network via 802.11b/g
LOOP:                ' check if initialization is done
    IF GET_NET_STATUS(7)=0 THEN
        GOTO LOOP
    END IF
TCP_OPEN(4,"192.168.6.24", 0, 21, 2, 59) 'log on to the ftp server
```

10.2.2 COMMAND: NCLOSE

NCLOSE	8000, 8200, 8300, 8400, 8700
---------------	-------------------------------------

Purpose To close an FTP connection.

Syntax NCLOSE(*N%*)

Remarks "*N%*" is "4" for FTP connection and "5" for Bluetooth FTP connection.

N%	Meaning
4	FTP connection (currently supported on 8000, 8200, 8300, 8400, 8700 only)
5	Bluetooth FTP connection (currently supported on 8200 only)

Example

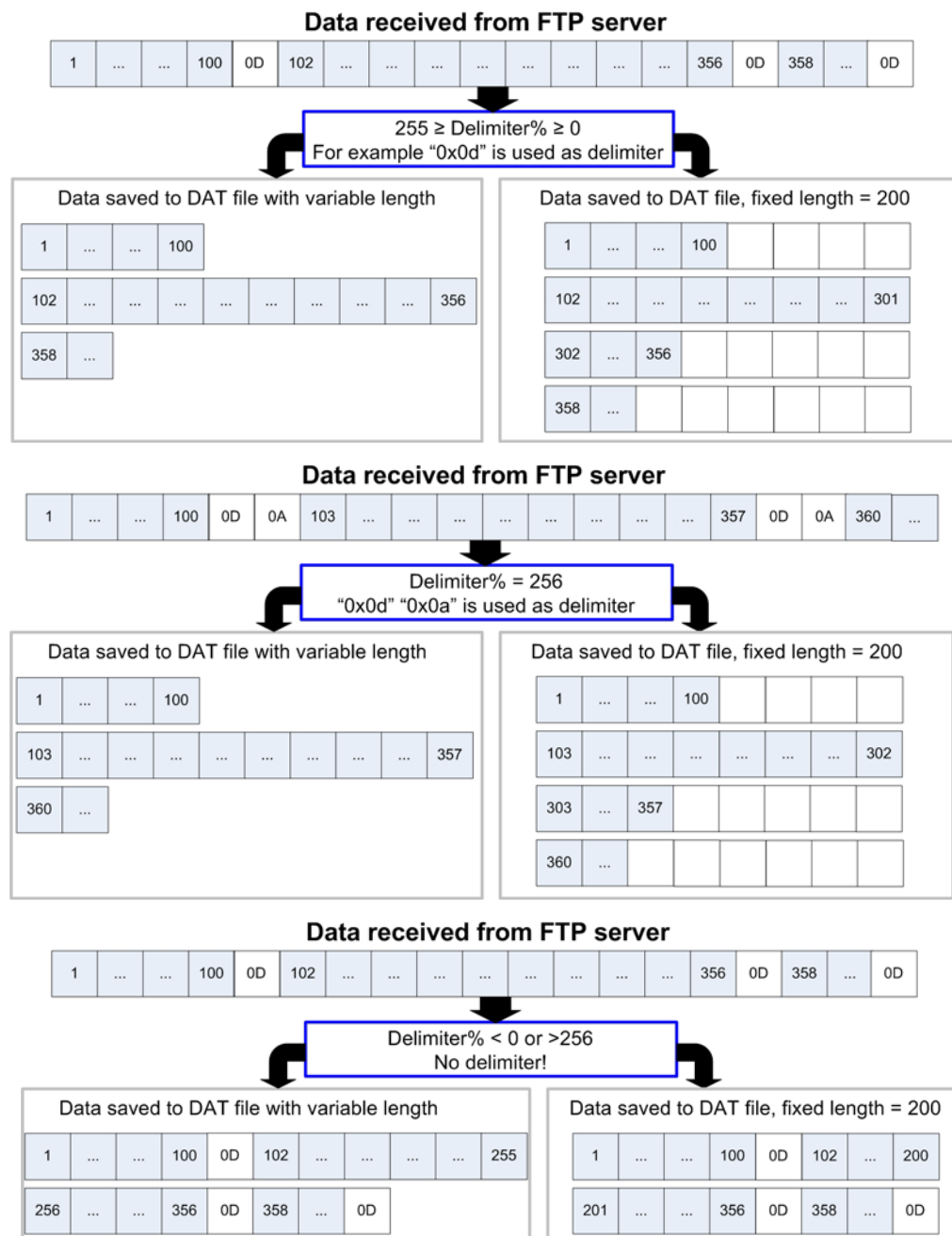
NCLOSE(4)	` close FTP connection
NCLOSE(5)	` close Bluetooth FTP connection

10.2.3 DELIMITER HANDLING

The delimiter set by **TCP_OPEN()** will affect the arrangement of data, which is either received in the DAT file system or stored in the buffer for being sent out over the network.

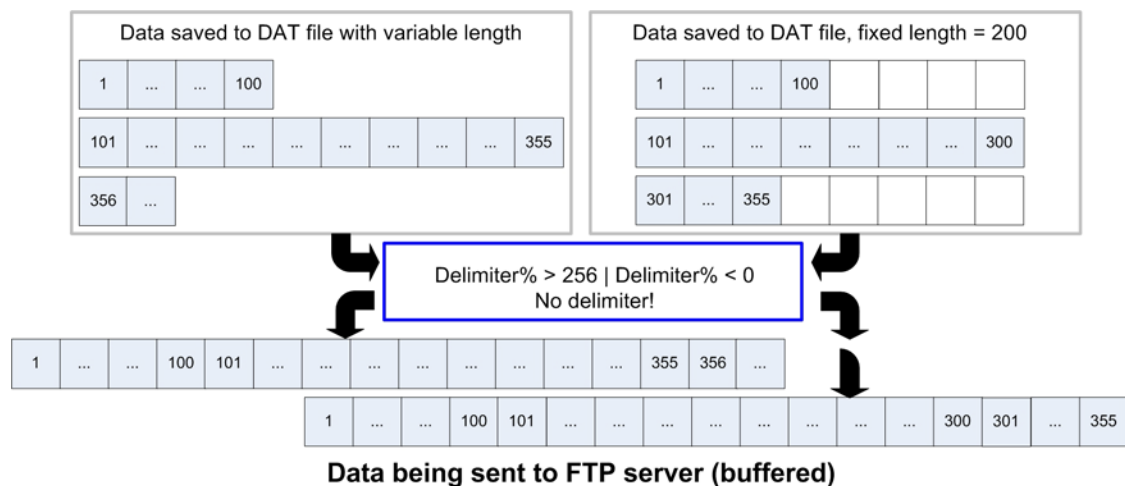
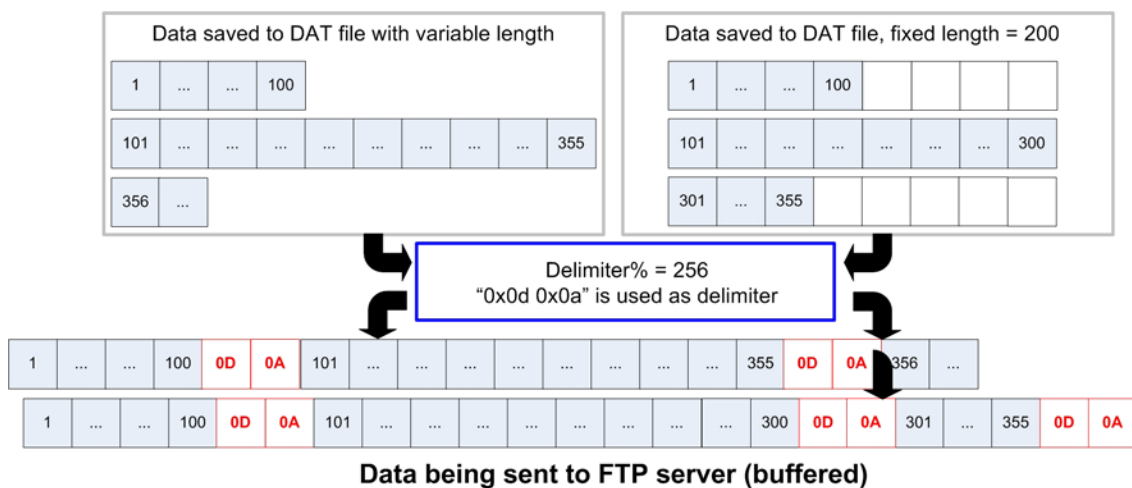
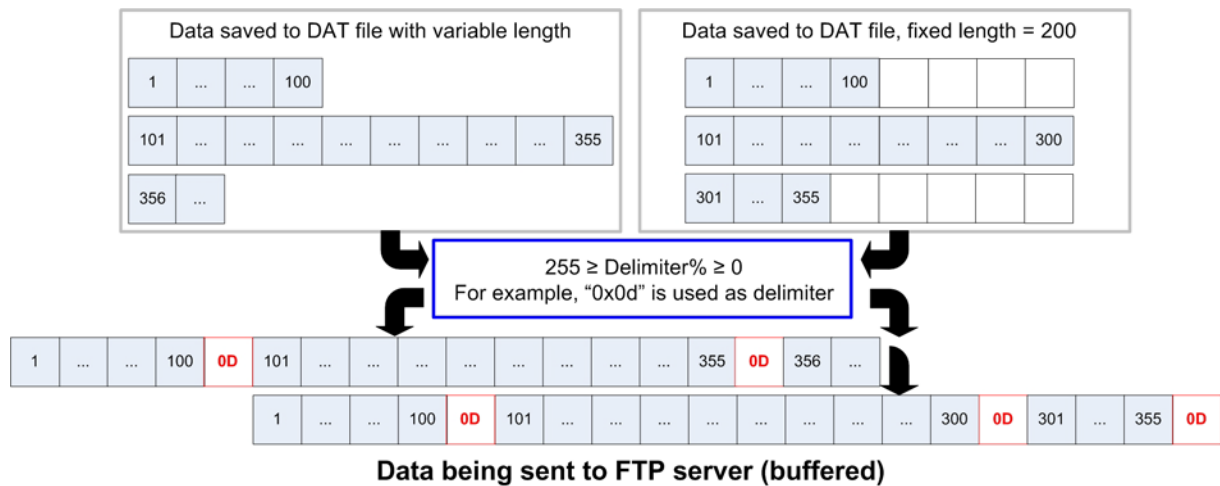
- ▶ The parameter for delimiter is optional and it is set to 0x0d (Carriage Return) by default. The valid values range from 0 to 256. If it is set to a value smaller than 0 or larger than 256, it will not transmit any delimiter.

Data received from Server



Send Data to Server

When sending a file to the FTP server, each line reading from the file system may be appended with a delimiter if specified, and put in the buffer.



10.3 DO FTP TASK

FTPRoutine\$ allows the mobile computers to log in to an FTP server and log out. The mobile computer can issue commands to the server to perform a specific task, such as create, change or remove directories on the server, delete, upload or download files, etc.

All these FTP tasks have been integrated in a BASIC function called **FTPRoutine\$**. Logging in must be carried out after having established a connection by calling **START TCPIP** and **TCP_OPEN**, while logging out must be carried out before the connection is terminated by **NCLOSE** and **STOP TCPIP**.

10.3.1 COMMAND: FTP_ROUTINE\$

Varying by the system running on the host, you may specify a relative path or absolute path when manipulating files or directories. For changing working directory, it allows the usage of ".." (back to the parent directory of the current director) and "/" (back to the root directory).

FTP_ROUTINE\$	8000, 8200, 8300, 8400, 8700
----------------------	-------------------------------------

Purpose To execute a specific FTP task.

Syntax A\$ = FTP_ROUTINE\$(N%, file%, Para1\$, Para2\$)

Remarks **Note that this function must be called after calling START TCPIP and TCP_OPEN.**

"A\$" is a string variable, indicating the status of a specific FTP task.

"N%" is an integer variable, indicating which FTP task is to be executed.

"file%" is an integer variable in the range of 1 to 6, indicating which file to access.

"Para1\$" is a string variable, indicating the first parameter of a specific FTP task.

"Para2\$" is a string variable, indicating the second parameter of a specific FTP task.

FTP Task	N%	FILE%	PARA1\$	PARA2\$
Get Directory	13	1~6	---	---
Change Directory	17	---	Path	---
Upload File	18	0: SD card access 1~6: DAT file 11~15: DBF file	Remote file name	Local file name (SD)
Append to File	19	0: SD card access 1~6: DAT file 11~15: DBF file	Remote file name	Local file name (SD)
Download File	20	0: SD card access 1~6: DAT file 11: DBF file 18: BASIC application (.tkn) 19: BASIC runtime (.bin)	Remote file name	Local file name (SD)
Rename FTP files (for 8200/8400/8700)	21	---	New remote file name	Old remote file name
Delete FTP files (for 8200/8400/8700)	22	---	Remote file name	---

Note: (1) "---" means the parameter can be ignored or is not required.

(2) For the FILE% marked with a sequence of hyphens "---", it must be

set to 0.

- (3) Append to File is not supported by Bluetooth FTP. Setting N% to 19 will get the same result as setting N% to 18.
- (4) When N% is either 18, 19, or 20 and File% is other than 0, PARA2\$ must be set to "". For examples:

FTP_RECV:

```
RESULT$=FTP_ROUTINE$(20,6, Remote$, "")
```

FTP_SEND:

```
RESULT$=FTP_ROUTINE$(18,1, Remote$, "")
```

Example

FTP_CWD:

```
RESULT$=FTP_ROUTINE$(17,0,"FTPTTest")
```

Remote\$="xact.txt"

Local\$="A:/Basic/Five/Basic/8400.txt"

FTP_RECV:

```
RESULT$=FTP_ROUTINE$(20,6, Remote$, "")
```

```
RESULT$=FTP_ROUTINE$(20,0, Remote$, Local$)
```

FTP_SEND:

```
RESULT$=FTP_ROUTINE$(18,1, Remote$, "")
```

FTP_RENAME:

```
RESULT$=FTP_ROUTINE$(21,0, NewRemote$, OldRemote$)
```

FTP_DELETE:

```
RESULT$=FTP_ROUTINE$(22,0, Remote$, "")
```

10.4 DOWNLOAD PROGRAM UPDATES

One of the major benefits of establishing an FTP connection is to download updates from the host for BASIC programs.

Use **FTP_ROUTINE\$(20, ...)** to receive the program files and **UPDATE_BASIC()** to activate each of them. Refer to [10.3 Do FTP Task](#) and [10.4.3 Activating Programs](#).

Note: For 8200/8400/8700, access to SD card is allowed. Refer to [10.6 SD Card Access](#).

10.4.1 UPDATING BASIC RUNTIME

Format

```
FTP_ROUTINE$(20, 19, RemoteFileName$, LocalFileName$)
```

```
/* Source file saved in SRAM */
```

```
FTP_ROUTINE$(20, (40~59), RemoteFileName$, LocalFileName$)
```

```
/* Source file saved on SD card, for 8200/8400/8700 only */
```

Example

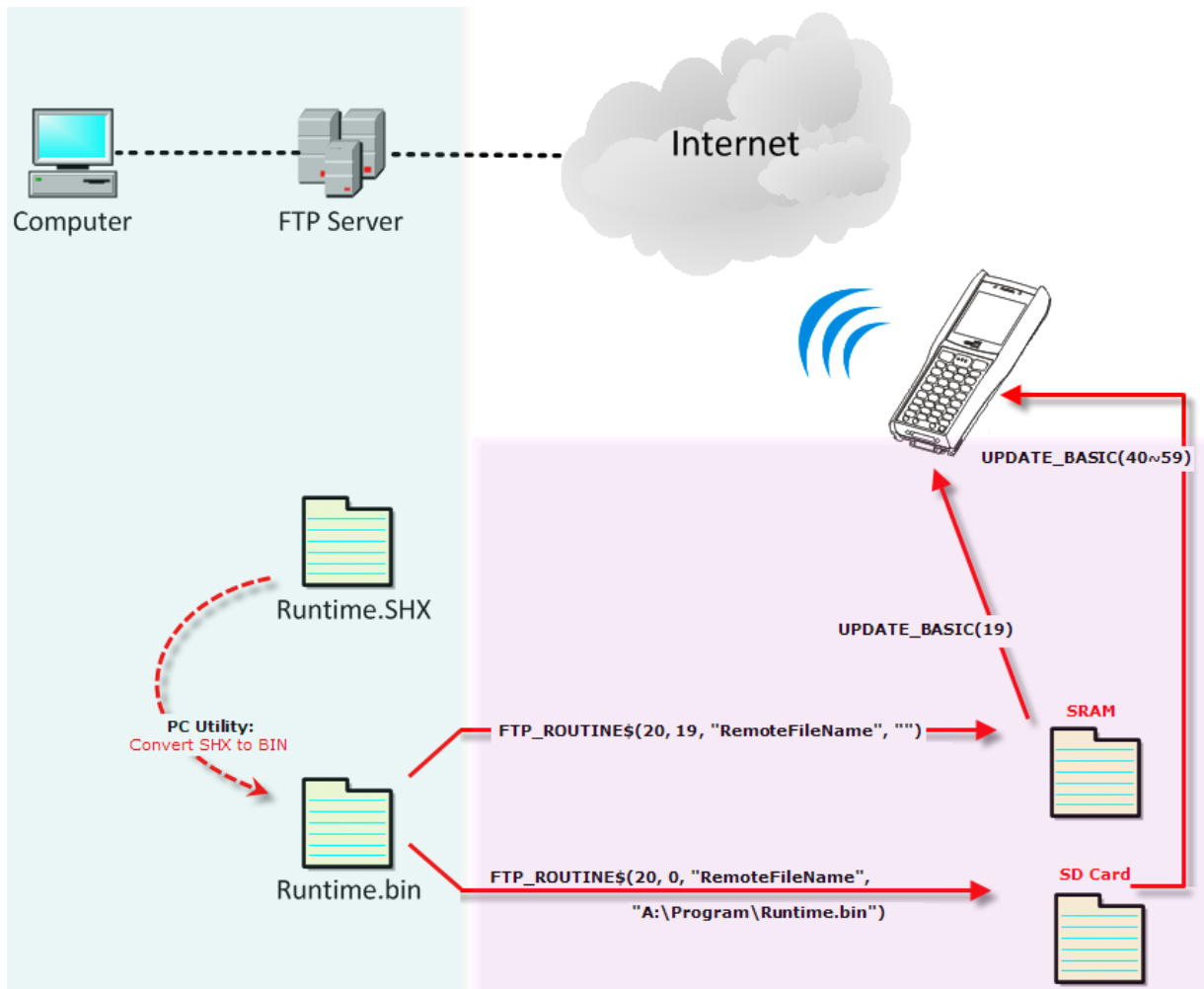
```
Remote$="basic.bin"
```

```
RESULT$=FTP_ROUTINE$(20, 19, REMOTE$, "")
```

```
UPDATE_BASIC(19)
```

```
...
```

Note: BASIC runtime program can be a .shx or .bin file. However, the file made available on the host must be a .bin file. Use PC utility "SHX2Bin.exe" to convert the program (.shx → .bin).



10.4.2 UPDATING BASIC APPLICATION

Format

```
FTP_ROUTINE$(20, 18, RemoteFileName$, LocalFileName$)

/* Source file saved in SRAM */

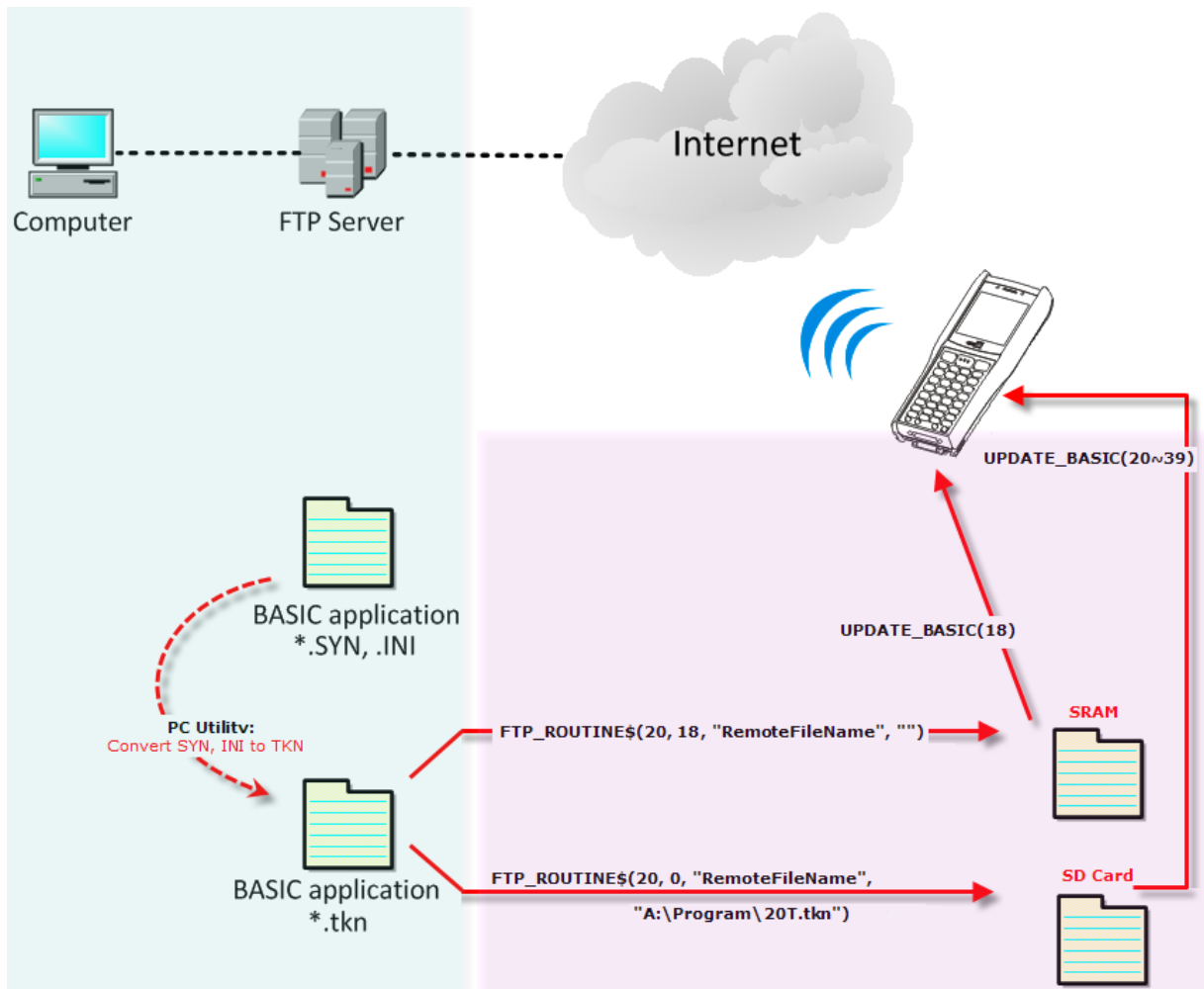
FTP_ROUTINE$(20, (20~39), RemoteFileName$, LocalFileName$)

/* Source file saved on SD card, for 8200/8400/8700 only */
```

Example

```
Remote$="graphic.tkn"
RESULT$=FTP_ROUTINE$(20, 18, REMOTE$, "")
UPDATE_BASIC(18)
...
```

Note: BASIC application program can be .syn, .ini, or a merged file (.tkn). However, the file made available on the host must be a .tkn file. Use PC utility "IniSyn2Token.exe" to merge the program (.syn, .ini → .tkn).



10.4.3 ACTIVATING PROGRAMS

If the source files have been downloaded to SRAM via FTP, use the following commands:

- ▶ **UPDATE_BASIC(18)** to activate an application program (.tkn)
- ▶ **UPDATE_BASIC(19)** to activate a runtime program (.bin)

If the source files have been downloaded to SD card via FTP, use the following commands:

- ▶ **UPDATE_BASIC(20~39)** to activate an application program (.tkn)
- ▶ **UPDATE_BASIC(40~59)** to activate a runtime program (.bin)

10.4.4 COMMAND: UPDATE_BASIC

UPDATE_BASIC

Purpose To have a BASIC program become the active program.

Syntax A% = UPDATE_BASIC(*file%*)

Remarks "A%" is an integer variable to be assigned to the result.

Value	Meaning
-1	Invalid file number
-2	Invalid file format
-8	No free space in flash before writing
-9	Fail to read program header (.ini)
-10 ^{Note}	Fail to read object file (.syn)
-11	RAM size cannot fit.
-12 ^{Note}	Fail to write new program into flash due to insufficient space, illegal address or the sector of flash cannot be erased.
-13 ^{Note}	Fail to write program header after new program written into flash
-14	Cannot find file on SD card
-15	Cannot read file on SD card
-16	File on SD card with filename length over 64 bytes

Note that it may not return the error code if the original BASIC program has been overwritten.

"*file%*" is an integer variable, indicating from which transaction file (or invisible file on 8200/8400/8700) the program is copied to the active area in flash memory. If successful, it will restart automatically.

Value	Meaning
1~6	Application program saved in file system ▶ Source file will be kept unless you erase it manually.

18	Application program (.tkn) saved in SRAM via FTP or DOWNLOAD_BASIC(18) ▶ Source file will be removed after execution. (currently supported on 8000, 8200, 8300, 8400, 8700 only)
19	Runtime program (.bin) saved in SRAM via FTP ▶ Source file will be removed after execution, but file system will be kept. (currently supported on 8000, 8200, 8300, 8400, 8700 only)
20~39	Application program (.tkn, or .syn, .ini) saved on SD card ▶ Source file will be kept after execution.
40~59	Runtime program(.bin or .shx) saved on SD card ▶ Source file and file system will be kept after execution.

- ▶ For 8200/8400/8700, if the source file is on SD card, "*file%*" must be set in a specific range, as shown above. You must follow these steps to make it active —

Step 1:	Rename the program by prefixing a number in the specific range. For example, EchoTest.ini -> 25EchoTest.ini EchoTest.syn -> 25EchoTest.syn
Step 2:	Copy the header file and object file to the specified directory "\Program" on SD card.
Step 3:	Call UPDATE_BASIC(25). System will search the file whose name starts with "25" in the directory "\Program".

Example

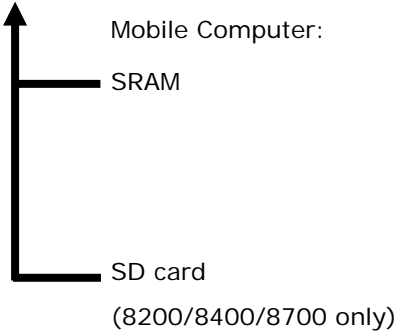
```
Error_Code% = UPDATE_BASIC(18)
```

10.5 FILE HANDLING

10.5.1 DAT FILES

Upload via FTP

Host



Mobile Computer:

SRAM

SD card

(8200/8400/8700 only)

Pre-processing of File in Format, Data, etc.

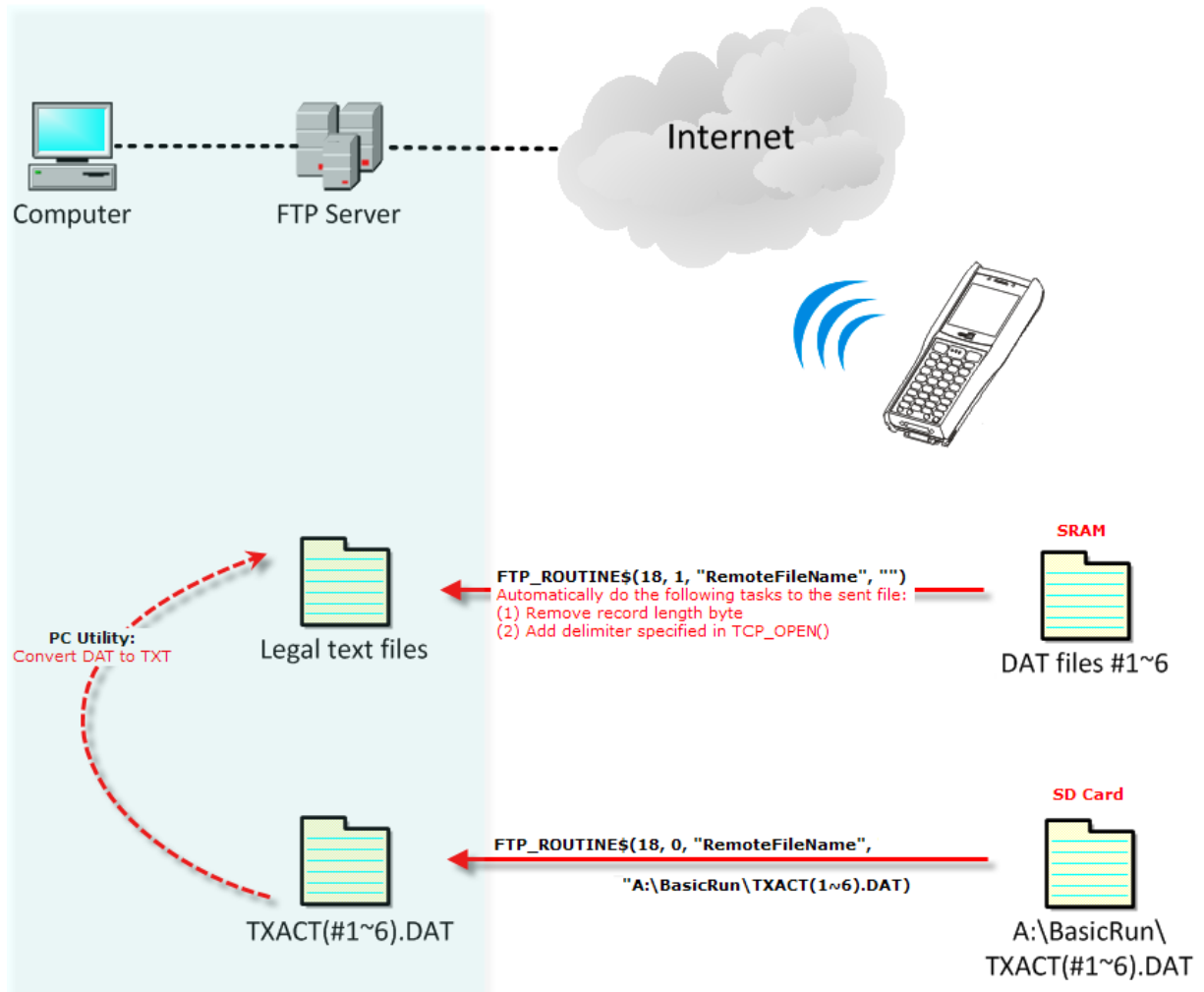
Not required

- ▶ DAT files will be uploaded as text files after automatically removing record length byte and adding desired delimiter specified in TCP_OPEN().

Remote only:

Use PC utility "DataConverter.exe" to convert TXACT.DAT files to text files .

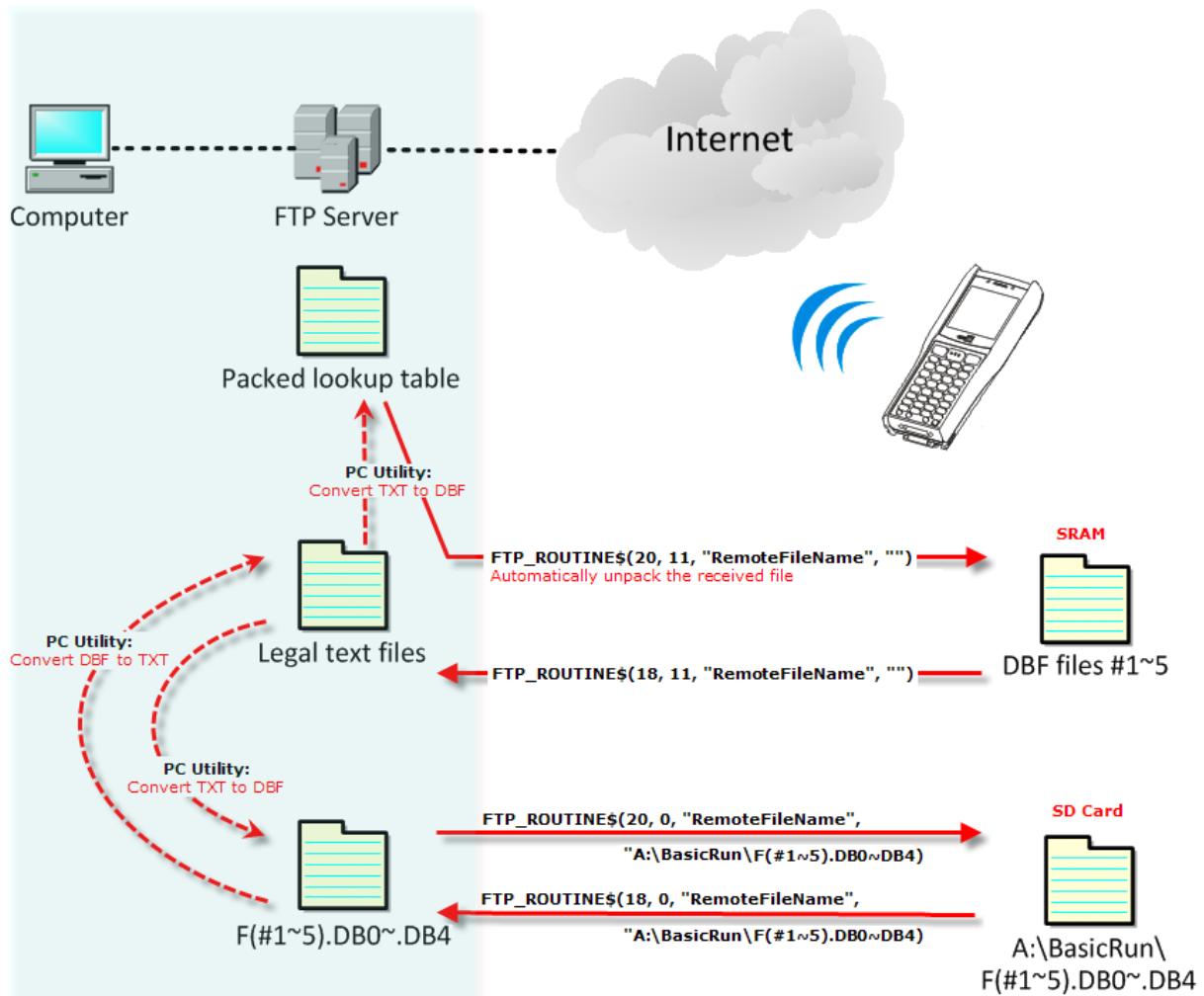
Note: For BASIC programs, 8200/8400/8700 with SD card cannot be treated as mass storage.



10.5.2 DBF FILES

Download via FTP		Pre-processing of File in Format, Data, etc.	
<div>Host</div> <div><div>Mobile Computer:</div><div>→ SRAM</div><div>→ SD card (8200/8400/8700 only)</div></div>	Remote only:	Use PC utility "DataConverter.exe" to create legal files (pack lookup table), which will then be automatically unpacked by the runtime program pre-loaded on the mobile computer.	
	Remote only:	Use PC utility "DataConverter.exe" to convert text files to SD files (DB0; DB1~4 for IDX files).	
Upload via FTP		Pre-processing of File in Format, Data, etc.	
<div>Host</div> <div><div>Mobile Computer:</div><div>→ SRAM</div><div>→ SD card (8200/8400/8700 only)</div></div>	Not required		
	Remote only:	Use PC utility "DataConverter.exe" to convert SD files (DB0; DB1~4 for IDX files) to text files.	

Note: For BASIC programs, 8200/8400/8700 with SD card cannot be treated as mass storage.



10.6 SD CARD ACCESS

For 8200/8400/8700, access to SD card is allowed. When a file name is required as an argument passed to a function call, it must be given in full path as shown below. Only absolute path is supported, and the file name is not case-sensitive.

- ▶ DAT files created on SD card by previous BASIC runtime are not compatible in file format with new BASIC runtime, starting from version 1.10.
- ▶ The size of DAT files on SD card can be calibrated via System Menu. If the function `DEL_TRANSACTION_DATA()` or `DEL_TRANSACTION_DATA_EX()` is called in BASIC applications to remove records from file top, the space will not be released immediately. Users have to refresh the size of "A:\BASICRUN\TXACTn.DAT" (n=1~6) via **System Menu | 1. SD Card Menu | 2. Access SD Card | 4. Check File Size**.

Warning: Although file name may be case-sensitive on remote host, for use with SD card, it is suggested to avoid using letter case for identifying two files with identical file name, such as "AAA.txt" and "aaa.txt".

10.6.1 DIRECTORY

Unlike the file system on SRAM, the file system on SD card supports hierarchical tree directory structure and allows creating sub-directories. Several directories are reserved for particular use.

Reserved Directory	Related Application or Function	Remark																																																					
\Program	<ul style="list-style-type: none">▶ Program Manager Download▶ Program Manager Activate▶ Kernel Menu Load Program▶ Kernel Menu Kernel Update▶ UPDATE_BASIC()	Store programs to this folder so that you can download them to 8200/8400/8700: <ul style="list-style-type: none">▶ C program — *.SHX▶ BASIC program — *.INI and *.SYN																																																					
\BasicRun	BASIC Runtime	<div>Store DAT and DBF files that are created and accessed in BASIC runtime to this folder. Their permanent filenames are as follows:</div> <table><tr><td colspan="3">DAT Filename</td></tr><tr><td>DAT file #1</td><td colspan="2">TXACT1.DAT</td></tr><tr><td>DAT file #2</td><td colspan="2">TXACT2.DAT</td></tr><tr><td>DAT file #3</td><td colspan="2">TXACT3.DAT</td></tr><tr><td>DAT file #4</td><td colspan="2">TXACT4.DAT</td></tr><tr><td>DAT file #5</td><td colspan="2">TXACT5.DAT</td></tr><tr><td>DAT file #6</td><td colspan="2">TXACT6.DAT</td></tr><tr><td colspan="3">DBF Filename</td></tr><tr><td rowspan="5">DBF file #1</td><td>Record file</td><td>F1.DB0</td></tr><tr><td>System Default Index</td><td>F1.DB1</td></tr><tr><td>Index file #1</td><td>F1.DB2</td></tr><tr><td>Index file #2</td><td>F1.DB3</td></tr><tr><td>Index file #3</td><td>F1.DB4</td></tr><tr><td rowspan="5">DBF file #2</td><td>Record file</td><td>F2.DB0</td></tr><tr><td>System Default Index</td><td>F2.DB1</td></tr><tr><td>Index file #1</td><td>F2.DB2</td></tr><tr><td>Index file #2</td><td>F2.DB3</td></tr><tr><td>Index file #3</td><td>F2.DB4</td></tr><tr><td rowspan="3">DBF file #3</td><td>Record file</td><td>F3.DB0</td></tr><tr><td>System Default Index</td><td>F3.DB1</td></tr><tr><td>Index file #1</td><td>F3.DB2</td></tr></table>	DAT Filename			DAT file #1	TXACT1.DAT		DAT file #2	TXACT2.DAT		DAT file #3	TXACT3.DAT		DAT file #4	TXACT4.DAT		DAT file #5	TXACT5.DAT		DAT file #6	TXACT6.DAT		DBF Filename			DBF file #1	Record file	F1.DB0	System Default Index	F1.DB1	Index file #1	F1.DB2	Index file #2	F1.DB3	Index file #3	F1.DB4	DBF file #2	Record file	F2.DB0	System Default Index	F2.DB1	Index file #1	F2.DB2	Index file #2	F2.DB3	Index file #3	F2.DB4	DBF file #3	Record file	F3.DB0	System Default Index	F3.DB1	Index file #1	F3.DB2
DAT Filename																																																							
DAT file #1	TXACT1.DAT																																																						
DAT file #2	TXACT2.DAT																																																						
DAT file #3	TXACT3.DAT																																																						
DAT file #4	TXACT4.DAT																																																						
DAT file #5	TXACT5.DAT																																																						
DAT file #6	TXACT6.DAT																																																						
DBF Filename																																																							
DBF file #1	Record file	F1.DB0																																																					
	System Default Index	F1.DB1																																																					
	Index file #1	F1.DB2																																																					
	Index file #2	F1.DB3																																																					
	Index file #3	F1.DB4																																																					
DBF file #2	Record file	F2.DB0																																																					
	System Default Index	F2.DB1																																																					
	Index file #1	F2.DB2																																																					
	Index file #2	F2.DB3																																																					
	Index file #3	F2.DB4																																																					
DBF file #3	Record file	F3.DB0																																																					
	System Default Index	F3.DB1																																																					
	Index file #1	F3.DB2																																																					

			Index file #2	F3.DB3
			Index file #3	F3.DB4
		DBF file #4	Record file	F4.DB0
			System Default Index	F4.DB1
			Index file #1	F4.DB2
			Index file #2	F4.DB3
			Index file #3	F4.DB4
		DBF file #5	Record file	F5.DB0
			System Default Index	F5.DB1
			Index file #1	F5.DB2
			Index file #2	F5.DB3
			Index file #3	F5.DB4
\AG\DBF \AG\DAT \AG\EXPORT \AG\IMPORT	Application Generator (a.k.a. AG)	Store DAT, DBF, and Lookup files that are created and/or accessed in Application Generator to this folder.		

10.6.2 FILE NAME

A file name must follow 8.3 format (= short filenames) — at most 8 characters for filename, and at most three characters for filename extension. The following characters are unacceptable: " * + , : ; < = > ? | []

- ▶ On 8200/8400/8700 Series, it can only display a filename of 1 ~ 8 characters (the null character not included), and filename extension will be displayed if provided. If a file name specified is longer than eight characters, it will be truncated to eight characters.
- ▶ Long filenames, at most 255 characters, are allowed when using 8200/8400/8700 equipped with SD card as a mass storage device. For example, you may have a filename "123456789.txt" created from your computer. However, when the same file is directly accessed on 8200/8400/8700, the filename will be truncated to "123456~1.txt".
- ▶ If a file name is specified other in ASCII characters, in order for 8200/8400/8700 to display it correctly, you may need to download a matching font file to 8200/8400/8700 first.
- ▶ The file name is not case-sensitive.

CRADLE COMMANDS

Through programming 8000/8300/8500 Series mobile computer, you can use cradle commands to control the Cradle.

- ▶ To determine whether cradle commands are applied, use the parameter *Parity%* of **SET_COM**.
- ▶ To determine which type of cradle to control, use the parameter *Baudrate%* of **SET_COM**.

Command	Parameters	Values	Remarks
SET_COM (N%, Baudrate%, Parity%, Data%, Handshake%)	N%	1 or 2	Indicates which COM port is to be set.
	Baudrate%	1: 115200 bps 3: 57600 bps 4: 38400 bps 5: 19200 bps 6: 9600 bps	<ul style="list-style-type: none"> ▶ Unless you have changed the baud rate setting via the DIP switch onboard, assign 1 for Ethernet Cradle because its factory setting is 115200 bps. ▶ Unless you have changed the baud rate setting via the DIP switch onboard, assign 3 for Modem Cradle because its factory setting is 57600 bps.
	Parity%	1: None 2: Odd 3: Even 4: Cradle commands	<p>The parity setting is NOT applicable.</p> <ul style="list-style-type: none"> ▶ Simply assign 4.
	Data%	1: 7 data bits 2: 8 data bits	<p>The data bits setting must be 8 bits.</p> <ul style="list-style-type: none"> ▶ Assign 2.
	Handshake%	1: None 2: CTS/RTS 3: XON/XOFF 4: Wedge	<p>The handshake setting is NOT applicable.</p> <ul style="list-style-type: none"> ▶ Simply assign 1.

For example,

- ▶ Call **SET_COM_TYPE(1, 3)** to set COM1 to Serial IR communication.
- ▶ To enable the issuing of cradle commands over COM port to the Ethernet Cradle, call **SET_COM(1, 1, 4, 2, 1)**;
to enable the issuing of cradle commands over COM port to the Modem Cradle, call **SET_COM(1, 3, 4, 2, 1)**.

- ▶ Call **OPEN_COM(1)** to initialize the connection over COM1.
- ▶ Issue a cradle command listed below. For example,
Sendbase\$ = "#vErSiOn?"
- ▶ Call **WRITE_COM(1, Sendbase\$)**.

...

Note: (1) Make sure the default delimiter 0x0d (CR) is in use; otherwise, call COM_DELIMITER().
(2) Unless you have changed the baud rate setting via the DIP switch onboard, pass the factory setting BAUD_115200 for Ethernet Cradle and BAUD_57600 for Modem Cradle.
(3) Baud rate will be reset to the DIP switch setting whenever you plug or unplug the RS-232 cable.

#fOrMaT:x	Cradle Command										
Purpose	To change the serial port settings of the cradle.										
Syntax	WRITE_COM(N%, A\$)										
Remarks	<p>"N%" is an integer variable, indicating which COM port the data is to be sent to.</p> <p>"A\$" is a string variable, representing the string to be sent. The format of A\$ should be "#fOrMaT:x", and</p> <table border="1"> <thead> <tr> <th>#fOrMaT:x</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>Set serial port mode to 8, N, 1</td></tr> <tr> <td>1</td><td>Set serial port mode to 7, N, 2</td></tr> <tr> <td>2</td><td>Set serial port mode to 7, O, 2</td></tr> <tr> <td>3</td><td>Set serial port mode to 7, E, 2</td></tr> </tbody> </table>	#fOrMaT:x	Meaning	0	Set serial port mode to 8, N, 1	1	Set serial port mode to 7, N, 2	2	Set serial port mode to 7, O, 2	3	Set serial port mode to 7, E, 2
#fOrMaT:x	Meaning										
0	Set serial port mode to 8, N, 1										
1	Set serial port mode to 7, N, 2										
2	Set serial port mode to 7, O, 2										
3	Set serial port mode to 7, E, 2										
Example	<pre>SET_COM_TYPE(1, 3) SET_COM(1, 3, 4, 2, 1) OPEN_COM(1) COMMANDSTR\$ = "#fOrMaT:2" WRITE_COM(1, COMMANDSTR\$) // set to 7,0,2 mode</pre>										
Return Value	If successful, it returns "#DONE".										
Remarks	This cradle command is supported by firmware version 3.50 and later.										
See Also	#SeRIaI										

#mOdEm	Cradle Command
Purpose	To set the working mode of cradle to MODEM mode.
Syntax	WRITE_COM(N%, A\$)
Remarks	<p>"N%" is an integer variable, indicating which COM port the data is to be sent to.</p> <p>"A\$" is a string variable, representing the string to be sent. The format of A\$ should be "#mOdEm".</p>
Example	<pre>SET_COM_TYPE(1, 3) SET_COM(1, 3, 4, 2, 1) OPEN_COM(1) COMMANDSTR\$ = "#mOdEm" WRITE_COM(1, COMMANDSTR\$) // set to MODEM mode</pre>
Return Value	If successful, it returns "#DONE" .
Remarks	After issuing the command, the baud rate of the cradle will be reset to the DIP switch setting.

Note: For the Ethernet Cradle, this command **"#mOdEm"** actually means "to select Ethernet" because the modem board has been replaced by the Ethernet board.

#SeRIAl	Cradle Command
Purpose	To reset the serial port settings of the cradle to defaults.
Syntax	WRITE_COM(N%, A\$)
Remarks	<p>"N%" is an integer variable, indicating which COM port the data is to be sent to.</p> <p>"A\$" is a string variable, representing the string to be sent. The format of A\$ should be "#SeRIAl".</p>
Example	<pre>SET_COM_TYPE(1, 3) SET_COM(1, 3, 4, 2, 1) OPEN_COM(1) COMMANDSTR\$ = "#SeRIAl" WRITE_COM(1, COMMANDSTR\$) // set to default</pre>
Return Value	<p>If successful, it returns "#DONE".</p> <p>Otherwise, it returns "#CABLE!" to indicate no RS-232 cable is detected.</p>
Remarks	This cradle command is supported by firmware version 3.30 and later. It will reset the serial port settings to defaults - N, 8, 1; however, the baud rate depends on the current DIP switch setting (57600 bps by default).

Note: Baud rate will be reset to the DIP switch setting whenever you plug or unplug the RS-232 cable.

#vErSiOn?	Cradle Command
Purpose	To retrieve the version information of the IR board.
Syntax	WRITE_COM(N%, A\$)
Remarks	<p>"N%" is an integer variable, indicating which COM port the data is to be sent to.</p> <p>"A\$" is a string variable, representing the string to be sent. The format of A\$ should be "#vErSiOn?".</p>
Example	<pre>SET_COM_TYPE(1, 3) SET_COM(1, 3, 4, 2, 1) OPEN_COM(1) COMMANDSTR\$ = "#vErSiOn?" WRITE_COM(1, COMMANDSTR\$)</pre>
Return Value	If successful, it returns the firmware version. For example, "#Ver03.20"

Note: There will be no response if the IR board version is no later than v3.00!

UNKNOWN COMMAND

It simply returns "#NAK".

NET PARAMETERS BY INDEX

The number in a pair of square brackets indicates the length of a string, e.g. GPRS_AP [21] means the maximum length of the string for remote IP address is 21 characters.

WIRELESS NETWORKING

Index		Configuration Item	Default Setup String	802.11b only	802.11b/g
GET_NET_PARAMETER\$	SET_NET_PARAMETER				
0 ~ 3		REMOTE_IP (string)	Read only	✓	✓
-1	1	LOCAL_IP (string)	0.0.0.0	✓	✓
-2	2	SUBNET_MASK (string)	0.0.0.0	✓	✓
-3	3	DEFAULT_GATEWAY (string)	0.0.0.0	✓	✓
-4	4	DNS_SERVER (string)	0.0.0.0	✓	✓
-5	5	LOCAL_NAME [33]	S/N	✓	✓
-6	6	SS_ID [33]	---	✓	✓
-7 to -10	7 to 10	WEP_KEY_1~4 [14]	---	✓	✓
-11	11	DHCP_ENABLE	Enable	✓	✓
-12	12	AUTHEN_ENABLE	Open	✓	✓
-13	13	WEP_LEN	128 bits	✓	✓
-14	14	SYSTEM_SCALE	Medium	✓	✓
-15	15	DEFAULT_WEP_KEY	1	✓	✓
-16		DOMAIN_NAME [129]	Read only	✓	✓
-17	17	WEP_ENABLE	Disable	✓	✓
-18	18	EAP_ENABLE	Disable	✓	✓
-19	19	EAP_ID [33]	---	✓	✓
-20	20	EAP_PASSWORD [33]	---	✓	✓
-21	21	POWER_SAVE_ENABLE	Enable	✓	✓
-22	22	PREAMBLE	Long	✓	✓
-23		MAC_ID (string)	Read only	✓	✓
-30	30	ADHOC	Disable	✓	✓
-31		FIRMWARE_VERSION [4]	Read only	✓	✓
-33	33	WPA_ENABLE WPA_PSK_ENABLE	Disable	✓	✓

Index		Configuration Item	Default Setup String	802.11b only	802.11b/g
GET_NET_PARAMETER\$	SET_NET_PARAMETER				
-34	34	WPA_PASSPHRASE [64]	---	✓	✓
-35		BSSID (string)	---	✓	✓
-36	36	FIXED_BSSID (string)	---	✓	✓
-37	37	ROAM_TXRATE_11B	2 Mbps	✓	✓
-38	38	ROAM_TXRATE_11G	11 Mbps	✓	✓
-39	39	WPA2_PSK_ENABLE	Disable	✓	✓
-83	83	SCAN_TIME	0	---	✓
-84	84	PROFILE1	---	---	✓
-85	85	PROFILE2	---	---	✓
-86	86	PROFILE3	---	---	✓
-87	87	PROFILE4	---	---	✓
	88	APPLY_PROFILE1	---	---	✓
	89	APPLY_PROFILE2	---	---	✓
	90	APPLY_PROFILE3	---	---	✓
	91	APPLY_PROFILE4	---	---	✓
-92	92	SCAN_CHANNEL	1111111111 1111	---	✓
-93	93	SCAN_CHANNEL_TIME	100	---	✓
-94	94	ROAMING_RSSI_THRESH OLD	-70	---	✓
-95	95	ROAMING_RSSI_DELTA	5	---	✓
-96	96	ROAMING_PERIOD	5	---	✓

Note: (1) The parameters ROAM_TXRATE_11B and ROAM_TXRATE_11G only work with "Custom-TxRate" system scale. Roaming starts when the data transmission rate gets lower than the specified value.

(2) Indexes -92 ~ -96 (GET)/92 ~ 96 (SET) are available only for 8200.

BLUETOOTH SPP, FTP, DUN

Index		Configuration Item	Default Setup String	SPP	FTP	DUN
GET_NET_PARAMETER\$	SET_NET_PARAMETER					
-5	5	LOCAL_NAME [33]	S/N	✓	✓	✓
-24		BT_MACID (string)	Read only	✓	✓	✓
-25	25	BT_REMOTE_NAME [20]	---	✓	✓	✓
-26	26	BT_SECURITY	Disable	✓	✓	✓
-27	27	BT_PIN_CODE [16]	---	✓	✓	✓
-28	28	BT_BROADCAST_ON	Enable	✓	✓	✓
-29	29	BT_POWER_SAVE_ON	Enable	✓	✓	✓
-32	32	BT_GPRS_APNAME [20]				✓
-40 to -47	40 to 47	BT_FREQUENT_DEVICE 1~8	---	✓	✓	✓

Note: When Bluetooth security is enabled without providing a pre-set PIN code, dynamic input of PIN code is supported.

GSM/GPRS

Index		Configuration Item	Default Setup String	GSM	GPRS
GET_NET_PARAMETER\$	SET_NET_PARAMETER				
-60		GSM_SERVICE_CENTER [21]	Read only	✓	
-61	61	GSM_PIN_CODE [9]	---	✓	✓
-62	62	GPRS_AP [21]	---		✓
-63		GSM_NET [21]	Read only	✓	
-64	64	GSM_MODEM_DIAL_NUM [21]	---	✓	
-65	65	GPRS_CHAP_ENABLE	Disable		✓
-66	66	GPRS_CHAP_PASSWORD [33]	---		✓
-67	67	GPRS_CHAP_USERNAME [33]	---		✓

Index		Configuration Item	Default Setup String	GPRS Cradle Transparent Mode
GET_NET_PARAMETER\$	SET_NET_PARAMETER			
-63		GSM_NET [21]	Read only	✓
-65	65	GPRS_CHAP_ENABLE	Disable	✓
-66	66	GPRS_CHAP_PASSWORD [33]	---	✓
-67	67	GPRS_CHAP_USERNAME [33]	---	✓

PPP

Index		Configuration Item	Default Setup String	PPP
GET_NET_PARAMETER\$	SET_NET_PARAMETER			
-70	70	PPP_DIALUPPHONE [20]	---	✓
-71	71	PPP_LOGINNAME [41]	---	✓
-72	72	PPP_LOGINPASSWORD [20]	---	✓
-73	73	PPP_BAUDRATE (cf. SET_COM)	1	✓

USB

Index		Configuration Item	Default Setup String	USB
GET_NET_PARAMETER\$	SET_NET_PARAMETER			
-80	80	USB_VCOM_BY_SN	Disable	✓

FTP

Index		Configuration Item	Default Setup String	FTP
GET_NET_PARAMETER\$	SET_NET_PARAMETER			
-81	81	FTP_USERNAME [65]	---	✓
-82	82	FTP_PASSWORD [65]	---	✓

NET STATUS BY INDEX

WIRELESS NETWORKING

For 8000/8200/8300/8400/8700 with 802.11b/g module, we suggest using indexes 14~16 instead of indexes 2~4. For 8231 with 802.11b/g/n module, indexes 2~4 are not supported.

Index	Configuration Item	Return Value		802.11b only	802.11b/g	802.11b/g/n
						8231 Return value
1	WLAN_State: Connection state	0 1	Disabled Connected	✓	✓	✓
2	WLAN_Quality: Link quality	0 ~ 10 10 ~ 15 15 ~ 30 30 ~ 50 50 ~ 80	Very poor Poor Fair Good Very good	✓	✓	Not Supported
3	WLAN_Signal: Signal strength level	0 ~ 30 30 ~ 60 60 ~ 120	Weak Moderate Strong	✓	✓	Not Supported
4	WLAN_Noise: Noise Level	1 2 ~ 3 4 ~ 5	Weak Moderate Strong	✓	✓	Not Supported
5	WLAN_Channel: Current channel #	1 ~ 11		✓	✓	✓
6	WLAN_TxRate: Transmit rate	1 2 4 8 16 32 48 64 80	1 Mbps 2 Mbps 5.5 Mbps 11 Mbps 6 Mbps 9 Mbps 12 Mbps 18 Mbps 24 Mbps	✓	✓	✓ ^{Note}

		96 112 128	36 Mbps 48 Mbps 54 Mbps			
7	NET_IPReady: Mobile computer IP status	-1 0 1	Error ^{Note} Not ready Ready	✓	✓	✓
14	WLAN_SNR: Signal to Noise ratio (dB)	0 ~ 10 10 ~ 20 20 ~ 30 30 ~ 40 over 40	Very poor Poor Fair Good Very good		✓	✓
15	WLAN_RSSI: Received Signal Strength Indication (-dBm)	0 ~ 60 60 ~ 75 over 75	Strong Moderate Weak		✓	✓ ^{Note} 0~60: Strong -60~-75: Moderate <-75: Weak
16	WLAN_NOISEFLO OR: Noise floor (-dBm)	0 ~ 92 92 ~ 98 over 98	Strong Moderate Weak		✓	✓ ^{Note}

Note: (1) If GET_NET_STATUS(7) returns -1, it means an abnormal break occurs during PPP, DUN-GPRS, or GPRS connection. Such disconnection may be caused by the mobile computer being out of range, improperly turned off, etc.

(2) Indexes 14~16 are only valid for 8000/8200/8231/8300/8400/8700 with 802.11b/g or 802.11b/g/n module.

(3) For 8231, unlike 8230, the received TxRate value (index 6) is exactly the data rate without conversion. For instance, suppose the data rate is 54Mbps; 8231 gets the received TxRate of 54 while 8230 gets 128 that is to be converted according to the parameter table.

(4) WLAN_RSSI and WLAN_NOISEFLOOR are negative values in dBm.

BLUETOOTH SPP, FTP, DUN

DUN¹ refers to Bluetooth DUN for connecting a modem.

DUN² refers to Bluetooth DUN-GPRS for activating a mobile's GPRS.

Index	Configuration Item	Return Value		SPP	FTP	DUN ¹	DUN ²
GET_NET_STATUS							
7	NET_IPReady: Mobile computer IP status	-1 0 1	Error ^{Note} Not ready Ready				✓
8	BT_State: Connection state	0 1	Disabled Connected	✓	✓	✓	✓
9	BT_Signal: RSSI signal level	-10 to -6 -6 to 5 5 to 30	Weak Moderate Strong	✓	✓	✓	✓

Note: If GET_NET_STATUS(7) returns -1, it means an abnormal break occurs during PPP, DUN-GPRS, or GPRS connection. Such disconnection may be caused by the mobile computer being out of range, improperly turned off, etc.

GSM/GPRS

Index	Configuration Item	Return Value		GSM	GPRS
GET_NET_STATUS					
10	GSM_state: connection state	0	Disabled	✓	✓
		1	Connected		
11	GSM_RSSIQuality: RSSI signal level	0	-113 dbm or less	✓	✓
		1	-111 dbm		
		2	-109 dbm		
			
		(3 ~ 29)	(+2 dbm per increment)		
		30	-53 dbm		
		31	-51 dbm or greater		
12	GSM_PINstate: PIN code status	0	Disabled	✓	✓
		1	PIN code required		
13	GSM connection status	0	Transmission undergoing	✓	
		1	Transmission completed		

EXAMPLES

WLAN EXAMPLE (802.11b/g)

Configure Network Parameters

Generally, network configuration has to be done in advance by calling **GET_NET_PARAMETERS** and **SET_NET_PARAMETER**.

Initialize Networking Protocol Stack & Wireless Module

The wireless module, such as of 802.11b/g, Bluetooth or GSM/GPRS, will not be powered until START TCPIP is called.

<i>Mobile Computer</i>	<i>WLAN (802.11b/g)</i>	<i>GPRS</i>	<i>Bluetooth DUN-GPRS</i>	<i>PPP via RS-232</i>
8062	---	---	START TCPIP(3)	---
8071	START TCPIP	---	---	---
8230	START TCPIP START TCPIP(0)	---	START TCPIP(3)	START TCPIP(5)
8260	---	---	START TCPIP(3)	START TCPIP(5)
8330	START TCPIP START TCPIP(0)	---	START TCPIP(3)	START TCPIP(5)
8362	---	---	START TCPIP(3)	START TCPIP(5)
8370	START TCPIP	---	---	START TCPIP(5)
8400	---	---	START TCPIP(3)	START TCPIP(5)
8470	START TCPIP START TCPIP(0)	---	START TCPIP(3)	START TCPIP(5)
8500	---	---	START TCPIP(3)	---
8570	START TCPIP START TCPIP(0)	---	START TCPIP(3)	---
8700	---	---	START TCPIP(3)	---
8770	START TCPIP START TCPIP(0)	---	START TCPIP(3)	---
8790	START TCPIP START TCPIP(0)	START TCPIP(2)	START TCPIP(3)	---

Note: (1) For the use of Modem Cradle, use **START TCPIP(4)** for PPP via IR or direct connect.
(2) For the use of Ethernet Cradle, use **START TCPIP(6)** for Ethernet via IR or direct connect.

Check Network Status

The **START TCPIP** routine does the first stage of the initialization process, and it will generate a system task to finish the rest of the process. When **START TCPIP** returns, the initialization process might not have been done yet. Therefore, it is necessary for the application program to check whether the status is "IP is ready" by calling **GET_TCPIP_MESSAGE** or **GET_NET_STATUS** before it proceeds to perform any networking operations.

Note: In case of initialization error, such as an abnormal break during PPP, DUN-GPRS, or GPRS connection, **GET_NET_STATUS(7)** will return -1.

Once the initialization process is done, the network status can be retrieved from the system. It will be periodically updated by the system. The application program must explicitly call **GET_NET_STATUS** to get the latest status.

Open Connection

Before reading and writing to the remote host, a connection must be established (opened). Call **TCP_OPEN** to open a connection. The application program needs to define a connection number (0~3), so that it can identify a particular connection in subsequent calls to other TCP/IP stack routines.

It is necessary for the application to check whether the status of the particular connection is "connected" by calling **GET_TCPIP_MESSAGE** before it proceeds to perform any read/write operations. Once the value of 4013 is returned (= connection is dropped abnormally, say, the mobile computer is shut down accidentally or by the **AUTO_OFF** timer), user program has to specify its own handling method. For example, if you wish to reconnect, simply call **START TCPIP** again.

Transmit Data

SOCKET_CAN_SEND

Before sending data to the network, call **SOCKET_CAN_SEND** to check if there is enough buffer size to write out the data immediately. It also can be used to check if the data being sent is more than 4 packets when there is no response from the remote host. Then, call **NWRITE** to send data on the network.

SOCKET_HAS_DATA

Before receiving data from the network, call **SOCKET_HAS_DATA** to check if there is data in the buffer. Then, call **NREAD\$** to receive data on the network.

Note: In case of an abnormal break during PPP, DUN-GPRS, or GPRS connection, **GET_TCPIP_MESSAGE** will return 4013 while **GET_NET_STATUS(7)** will return -1.

Other Useful Functions...

There are other routines for obtaining additional information or setting control for a connection.

SOCKET_OPEN, SOCKET_HAS_DATA, etc.
To check the connection status by polling method.
GET_NET_PARAMETER\$
To get the networking configuration and the remote site IP address.
TCP_IP_ERR_CODE
To get the operation result after calling any TCPIP routines.
TCPIP Event Trigger
ON TCPIP GOSUB... and OFF TCPIP are used to get higher working performance. Once the TCPIP event occurs, it is necessary for the application program to check the trigger type by getting the value of the GET_TCPIP_MESSAGE routine.

Close Connection

Call **NCLOSE** to terminate a particular connection when the application program does not use it any more.

Terminate Networking Protocol Stack & Wireless Module

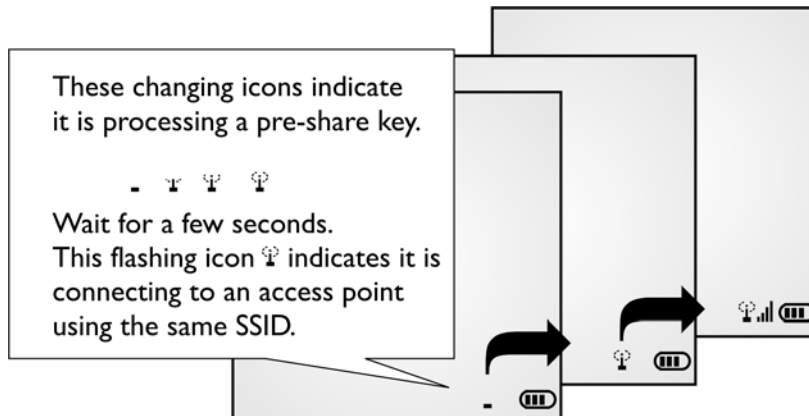
When the application program wishes to stop using the network, call **STOP TCPIP** to terminate networking and shut down the power to the module so that it can save power. To enable the network again, it is necessary to call **START TCPIP** again.

Note: After calling STOP TCPIP, any previous network connection and data will be lost.

WPA ENABLED FOR SECURITY

If WPA-PSK/WPA2-PSK is enabled for security, SSID and Passphrase will be processed to generate a pre-share key. If you change SSID or Passphrase, it will have to re-generate a pre-share key.

- 1) For initial association with an access point, you will see an antenna icon developing on the screen to indicate that the mobile computer is processing a pre-share key.



- 2) After having generated the pre-share key, the mobile computer proceeds to establish a connection with an access point, and you will see the whole antenna is flashing.
- 3) When the mobile computer has been connected to the access point successfully, you will see the whole antenna and the indication of wireless signal strength.

Note: Be aware that these icons will appear on the device screen after `START TCPIP()` is called. (WPA-PSK/WPA2-PSK must be enabled first!)

BLUETOOTH EXAMPLES

Command	Parameters	Values	Remarks
SET_COM (N%, Baudrate%, Parity%, Data%, Handshake%)	N%	2	Indicates Bluetooth COM port is to be set.
	Baudrate%	1: 115200 bps 2: 76800 bps 3: 57600 bps 4: 38400 bps 5: 19200 bps 6: 9600 bps 7: 4800 bps 8: 2400 bps	The baud rate setting is NOT applicable to Bluetooth. Simply assign – <ul style="list-style-type: none"> ▶ 1 for SPP Slave ▶ 4 for SPP Master ▶ 5 for DUN ▶ 6 for HID
	Parity%	1: None 2: Odd 3: Even 4: Cradle Commands	The parity setting is NOT applicable to Bluetooth. <ul style="list-style-type: none"> ▶ Simply assign 1 for Bluetooth. ▶ To determine which type of cradle to control, use the parameter <i>Baudrate%</i> of SET_COM. Refer to Appendix I — Cradle Commands.
	Data%	1: 7 data bits 2: 8 data bits	The data bits setting is NOT applicable to Bluetooth. <ul style="list-style-type: none"> ▶ Simply assign 1 for Bluetooth.
	Handshake%	1: None 2: CTS/RTS 3: XON/XOFF 4: Wedge Emulator	The handshake setting is NOT applicable to Bluetooth. Simply assign – <ul style="list-style-type: none"> ▶ 1 for Bluetooth SPP/DUN/HID ▶ 4 for Bluetooth Wedge Emulator

SPP MASTER

Inquiry

Call **BT_INQUIRY\$** to discover nearby Bluetooth devices.

Pairing

Call **BT_PAIRING (addr\$, 3)** to pair with a Bluetooth device.

Set Communication Type

Call **SET_COM_TYPE(2, 5)** to set COM2 for Bluetooth communication.

Set Bluetooth Service

Call **SET_COM(2, 4, 1, 1, 1)** to initialize Bluetooth SPP Master.

Open COM Port

Call **OPEN_COM(2)** to initialize the Bluetooth module and set up connection.

Check Connection

Call **GET_NET_STATUS(8)** to detect if connection is completed. For example,

LOOP003:

```
IF GET_NET_STATUS(8) = 0 THEN
    GOTO LOOP003
    BEEP(4400, 4)
    CLS
    PRINT "Connect OK"
```

Transmit/receive Data

Call **WRITE_COM(2)** and **READ_COM\$(2)** to transmit and receive data respectively.

Check Connection

Call **GET_NET_STATUS(8)** to detect if connection is maintained. For example,

```
IF GET_NET_STATUS(8) = 0 THEN
    BEEP(3300, 4)
    CLOSE_COM(2)
END IF
```

Close COM Port

Call **CLOSE_COM(2)** to terminate communication and shut down the Bluetooth module.

SPP SLAVE

Set Communication Type

Call **SET_COM_TYPE(2, 5)** to set COM2 for Bluetooth communication.

Set Bluetooth Service

Call **SET_COM(2, 1, 1, 1, 1)** to initialize Bluetooth SPP Slave.

Open COM Port

Call **OPEN_COM(2)** to initialize the Bluetooth module and set up connection.

Check Connection

Call **GET_NET_STATUS(8)** to detect if connection is completed. For example,

LOOP003:

```
IF GET_NET_STATUS(8) = 0 THEN
    GOTO LOOP003
    BEEP(4400, 4)
    CLS
    PRINT "Connect OK"
```

Transmit/receive Data

Call **WRITE_COM(2)** and **READ_COM\$(2)** to transmit and receive data respectively.

Check Connection

Call **GET_NET_STATUS(8)** to detect if connection is maintained. For example,

```
IF GET_NET_STATUS(8) = 0 THEN
    BEEP(3300, 4)
    CLOSE_COM(2)
END IF
```

Close COM Port

Call **CLOSE_COM(2)** to terminate communication and shut down the Bluetooth module.

WEDGE EMULATOR VIA SPP

Refer to **Part I: 4.9 Keyboard Wedge Commands** and **4.9.3 Wedge Emulator**.

SET_COM(N%, Baudrate%, Parity%, Data%, Handshake%) - To set the wedge emulation flag, use the last parameter regarding hardware handshake setting.

```
SET_COM_TYPE(2, 5)
```

```
SET_COM(2, 1, 1, 1, 4)
```

```
OPEN_COM(2)
```

And then, use the normal wedge functions to send data.

```
SET_COM_TYPE(2, 5)
```

```
SET_COM(2, 1, 1, 1, 4)
```

```
OPEN_COM(2)
```

```
CLS
```

```
PRINT "Wait to Connect"
```

```
LOOP000:
```

```
IF WEDGE_READY = 0 THEN GOTO LOOP000
```

```
BEEP(4400, 4)
```

```
CLS PRINT "OK! Try to Send"
```

```
LOOP:
```

```
KeyData$ = INKEY$
```

```
IF KeyData$ = "" THEN GOTO LOOP
```

```
IF KeyData$ = "0" THEN
```

```
    IF WEDGE_READY = 1 THEN
```

```
        PRINT "READY"
```

```
    ELSE
```

```
        PRINT "NOT READY"
```

```
    END IF
```

```
ELSE IF KeyData$ = "1" THEN
```

```
        SEND_WEDGE("Hello")  
ELSE IF KeyData$ = "2" THEN  
    PRINT "Hello"  
END IF  
GOTO LOOP
```

HID

Configure Wedge Settings

Bluetooth HID makes use of the **WedgeSettings\$** array to govern the HID operations. Refer to **Part I: 4.9 Keyboard Wedge Commands**.

Parameter	Bit	Description
Wedge_1\$	7 - 0	KBD / Terminal Type
Wedge_2\$	7	1: Enable capital lock auto-detection 0: Disable capital lock auto-detection
Wedge_2\$	6	1: Capital lock on 0: Capital lock off
Wedge_2\$	5	1: Ignore alphabets' case 0: Alphabets are case-sensitive
Wedge_2\$	4 - 3	00: Normal 10: Digits at lower position 11: Digits at upper position
Wedge_2\$	2 - 1	00: Normal 10: Capital lock keyboard 11: Shift lock keyboard
Wedge_2\$	0	1: Use numeric keypad to transmit digits 0: Use alpha-numeric key to transmit digits
Wedge_3\$	0	HID Character Transmit Mode 1: By character 0: Batch processing

Wedge_1\$: It is used to determine which type of keyboard wedge is applied, and the possible value is listed below.

Setting Value	Terminal Type	Setting Value	Terminal Type
0	Null (Data Not Transmitted)	8	PCAT (BE)
1	PCAT (US)	9	PCAT (SP)
2	PCAT (FR)	10	PCAT (PO)
3	PCAT (GR)	11	IBM A01-02 (Japanese OADG109)
4	PCAT (IT)	12	PCAT (Turkish)
5	PCAT (SV)	13	PCAT (Hungarian), 8200/8400/8700
6	PCAT (NO)	14	PCAT (Swiss(German)),8200/8400/8700
7	PCAT (UK)		

See **Wedge_2\$**: For details, refer to **Part I: 4.9 Keyboard Wedge Commands**.

Wedge_3\$: It is used to configure how it sends data to the host, either by character or batch processing.
Communication Type

Call **SET_COM_TYPE(2, 5)** to set COM2 for Bluetooth communication.

Set Bluetooth Service

Call **SET_COM(2, 6, 1, 1, 1)** to initialize Bluetooth HID functionality.

Open COM Port

Call **OPEN_COM(2)** to initialize the Bluetooth module and set up connection.

Check Connection

Call **GET_NET_STATUS(8)** to detect if connection is completed. For example,

LOOP003:

```
IF GET_NET_STATUS(8) = 0 THEN
    GOTO LOOP003
    BEEP(4400, 4)
    CLS
    PRINT "Connect OK"
```

Frequent Device List

When there is a host device recorded in the Frequent Device List, the mobile computer (as SPP Master) will automatically connect to it. If the connection fails, the mobile computer will try again. If it fails for the second time, the mobile computer will wait 7 seconds for another host to initiate a connection. If still no connection is established, the mobile computer will repeat the above operation.

When there is no device recorded in the Frequent Device List, the mobile computer (as SPP Slave) simply must wait for a host device (as SPP Master) to initiate a connection.

Note: As an HID input device (keyboard), the mobile computer must wait for a host to initiate a connection. Once the HID connection is established, the host device will be recorded in the Frequent Device List identified as HID Connection.

Transmit Data

Call **WRITE_COM(2, *data)** to transmit data.

Check Connection

Call **GET_NET_STATUS(8)** to detect if connection is maintained. For example,

```
IF GET_NET_STATUS(8) = 0 THEN
    BEEP(3300, 4)
    CLOSE_COM(2)
END IF
```

Close COM Port

Call **CLOSE_COM(2)** to terminate communication and shut down the Bluetooth module.

DUN

Inquiry

Call **BT_INQUIRY\$** to discover nearby Bluetooth devices.

Pairing

Call **BT_PAIRING (addr\$, 4)** to pair with a Bluetooth device that can work as a modem.

Set Communication Type

Call **SET_COM_TYPE(2, 5)** to set COM2 for Bluetooth communication.

Set Bluetooth Service

Call **SET_COM(2, 5, 1, 1, 1)** to initialize Bluetooth DUN functionality.

Open COM Port

Call **OPEN_COM(2)** to initialize the Bluetooth module and set up connection.

Check Connection

Call **GET_NET_STATUS(8)** to detect if connection is completed. For example,

LOOP003:

```
IF GET_NET_STATUS(8) = 0 THEN
    GOTO LOOP003
    BEEP(4400, 4)
    CLS
    PRINT "Connect OK"
```

Transmit/receive Data

Call **WRITE_COM(2)** and **READ_COM\$(2)** to transmit and receive data respectively.

Check Connection

Call **GET_NET_STATUS(8)** to detect if connection is maintained. For example,

```
IF GET_NET_STATUS(8) = 0 THEN
    BEEP(3300, 4)
    CLOSE_COM(2)
END IF
```

Close COM Port

Call **CLOSE_COM(2)** to terminate communication and shut down the Bluetooth module.

DUN-GPRS

To activate the GPRS functionality on a mobile phone via the built-in Bluetooth dial-up networking technology, follow the same programming flow of [WLAN Example \(802.11b/g\)](#).

- ▶ Before calling **START TCPIP(3)**, the following parameters of DUN-GPRS must be specified.

Index		Configuration Item	Default	Description
-32	32	P_ BT_GPRS_APNAME [20]	Null	Name of Access Point for Bluetooth DUN-GPRS

FTP (8200 ONLY)

Inquiry

Call **BT_INQUIRY\$** to discover nearby Bluetooth devices.

Pairing

Call **BT_PAIRING (addr\$, 7)** to pair with the FTP server.

Open Connection

Before transferring files with the FTP server, a connection must be established (opened). Call **TCP_OPEN (5, "0.0.0.0", 0, 0, 2, [, Delimiter%])** to open a connection.

Perform FTP Tasks

Call **FTP_ROUTINE\$ (N%, file%, Para1\$, Para2\$)** to execute a specific FTP task.

Close Connection

Call **NCLOSE (5)** to terminate the connection.

GSM/GPRS EXAMPLES

GPRS

To establish a connection to the content server connected to the internet, follow the same programming flow of [WLAN Example \(802.11b/g\)](#). Only client-initiated connection is supported.

Connecting Mobile Computer

Before calling **START TCPIP(2)**, the following parameters of GPRS must be specified.

Index		Configuration Item	Default	Description
-61	61	P_ GSM_PIN_CODE [9]	Null	PIN Code for GSM/GPRS
-62	62	P_ GPRS_AP [21]	Null	Name of Access Point for GPRS

Connecting 8400 GPRS Cradle (Transparent Mode)

Before calling **START TCPIP(7)**, use AT commands to configure PIN code and GPRS AP name.

- ▶ If CHAP is enabled, you must configure the settings from the mobile computer.
- ▶ It fails to initialize a connection in the following conditions: (1) PIN code and GPRS AP name are not configured correctly via AT commands, and (2) CHAP settings are not configured correctly on 8400.

Note: A client-initiated connection occurs when the connection is established in response to a request from the client.

GSM

Configure Parameters

Call **SET_NET_PARAMETER** to set variables, such as PINCode[], ModemDialNum[], and so on.

It is recommended that the correct PIN code should be initialized before opening the GSM port. This is because the PIN code will be taken as a password to activate the SIM card. Therefore, any input of incorrect PIN code during initialization will result in wasting one attempt of PIN entry. If you fail the PIN entry three times, the procedure of PIN code entry will be locked.

Set Communication Type

Call **SET_COM_TYPE(3, 6)** to set COM3 for SMS (GSM_SMS).






Or call **SET_COM_TYPE(3, 7)** to set COM3 for data call (GSM_Modem).

Open COM Port

Call **OPEN_COM(3)** to initialize the GSM/GPRS module. The initialization takes about 10 seconds.

An antenna icon representing the GSM(*GSM_SMS* only)/GPRS operation will be displayed, and it keeps flashing until **OPEN_COM(3)** procedure is completed. Once the procedure is completed, the signal strength bar will be displayed next to the antenna icon, and it will be updated every five seconds. The level of the signal strength bar ranges from 0 to 5.

- ▶ The value of the PIN code will be fetched as a password required for initializing the operation.
- ▶ Refer to [6.2.1 PIN Procedure](#) and [6.2.2 PUK Procedure](#) for handling PINCode[] errors. New PIN code re-entry and PUK unblock operation are furnished.
- ▶ Once the PIN code check is passed, PINCode[] will be updated with the input value.
- ▶ After **OPEN_COM(3)** is completed, relevant information will be obtained, such as SMSServiceCenter[], NET[], and PINstatus.

Signal Bar	RSSI Range	
(Empty)	$x < 10$	$(< -93 \text{ dbm})$
	$10 \leq x < 12$	$(-93 \leq x < -89 \text{ dbm})$
	$12 \leq x < 15$	$(-89 \leq x < -83 \text{ dbm})$
	$15 \leq x < 18$	$(-83 \leq x < -77 \text{ dbm})$
	$18 \leq x < 21$	$(-77 \leq x < -71 \text{ dbm})$
	$21 \leq x$	$(-71 \leq x)$

Note: For GSM_Modem, refer to GSMModemGetRSSI(). GetNetStatus(GSM_RSSIQuality) will become available only when GSMModemGetRSSI() is called first.

Check Connection

Call **GET_NET_STATUS(13)** to detect if the initialization is completed. For example,

```
OPEN_COM(3)
```

```
LOOP:
```

```
GSMeot% = GET_NET_STATUS(13)
```

```
IF GSMeot% = 1 THEN
```

```
    GOTO INIT_DONE
```

```
ENDIF
```

```
...
```

```
GOTO LOOP
```

```
INIT_DONE:
```

```
...
```

Such checking must be carried out to ensure the initialization of the GSM/GPRS module has been completed. **GET_NET_STATUS(13)** will return 1 if the initialization is completed.

Note: The POWER key will be disabled during the connection process. Yet, the [ESC] key is provided for being able to abort the PIN code check while connecting.

Transmit/receive Data

Call **WRITE_COM(3, A\$)** and **READ_COM\$(3)** to transmit and receive data respectively.

Check Connection

Call **GET_NET_STATUS(13)** to detect if the transmission is completed. For example,

```
WRITE_COM(3, A$)
```

```
W_LOOP:
```

```
    GSMeot% = GET_NET_STATUS(13)
```

```
    IF GSMeot% = 1 THEN
```

```
        GOTO W_DONE
```

```
    ENDIF
```

```
    ...
```

```
    GOTO W_LOOP
```

```
W_DONE:
```

```
...
```

Close COM Port

Call **CLOSE_COM(3)** to terminate communication and shut down the GSM/GPRS module.

USB EXAMPLE

USB VIRTUAL COM

Set Communication Type

Call **SET_COM_TYPE(5, 9)** to set COM5 for USB Virtual COM communication.

Open COM Port

Call **OPEN_COM(5)** to initialize the COM port.

Transmit/receive Data

Call **WRITE_COM(5, A\$)** and **READ_COM\$(5)** to transmit and receive data respectively.

Close COM Port

Call **CLOSE_COM(5)** to terminate USB communication.

USB HID

Configure Wedge Settings

Like Bluetooth HID, USB HID also makes use of the **WedgeSetting\$** array to govern the HID operations. Refer to **Part I: 4.9 Keyboard Wedge Commands**.

Parameter	Bit	Description
Wedge_1\$	7 – 0	KBD / Terminal Type
Wedge_2\$	7	1: Enable capital lock auto-detection 0: Disable capital lock auto-detection
Wedge_2\$	6	1: Capital lock on 0: Capital lock off
Wedge_2\$	5	1: Ignore alphabets' case 0: Alphabets are case-sensitive
Wedge_2\$	4 – 3	00: Normal 10: Digits at lower position 11: Digits at upper position
Wedge_2\$	2 – 1	00: Normal 10: Capital lock keyboard 11: Shift lock keyboard
Wedge_2\$	0	1: Use numeric keypad to transmit digits 0: Use alpha-numeric key to transmit digits
Wedge_3\$	0	HID Character Transmit Mode 1: By character 0: Batch processing

Wedge_1\$: It is used to determine which type of keyboard wedge is applied, and the possible value is listed below.

Setting Value	Terminal Type	Setting Value	Terminal Type
0	Null (Data Not Transmitted)	8	PCAT (BE)
1	PCAT (US)	9	PCAT (SP)
2	PCAT (FR)	10	PCAT (PO)
3	PCAT (GR)	11	IBM A01-02 (Japanese OADG109)
4	PCAT (IT)	12	PCAT (Turkish)
5	PCAT (SV)	13	PCAT (Hungarian), 8200/8400/8700
6	PCAT (NO)	14	PCAT (Swiss(German)),8200/8400/8700
7	PCAT (UK)		

Set **Wedge_2\$**: For details, refer to **Part I: 4.9 Keyboard Wedge Commands**.

Wedge_3\$: It is used to configure how it sends data to the host, either by character or batch processing. **Communication Type**

Call **SET_COM_TYPE(5, 8)** to set COM5 for USB HID communication.

Open COM Port

Call **OPEN_COM(5)** to initialize the COM port.

Transmit Data

Call **WRITE_COM(5, A\$)** to transmit data.

Close COM Port

Call **CLOSE_COM(5)** to terminate USB communication.

USB MASS STORAGE DEVICE

Set Communication Type

Call **SET_COM_TYPE(5, 10)** to set COM5 for the use of USB removable disk.

Open COM Port

Call **OPEN_COM(5)** to initialize the COM port.

Check Connection

Call **IOPIN_STATUS(3)** to detect if connection is completed. For example,

```
LOOP1:
    A%=IOPIN_STATUS(3)
IF A% = 0 THEN
    PRINT "Disconnect"
ELSE IF A% = 1 THEN
    PRINT "Connected"
ELSE IF A% = 3 THEN
    PRINT "Device is being accessed"
END IF
    GOTO LOOP1
```

FTP MESSAGE

FTP messages are responses to FTP commands, and each consists of a 4-digit response code ("5XYZ").

You may use **GET_TCPIP_MESSAGE()** to get the message after executing an FTP task:

```
TCP_EVENT% = GET_TCPIP_MESSAGE
```

TCP_EVENT%				Description
5				The 1 st digit is always "5".
	X			The 2 nd digit refers to which FTP task is executed. 1: Open a connection by TCP_OPEN() 2: Get directory by FTP_ROUTINE\$(13, ...) 3: Change directory by FTP_ROUTINE\$(17, ...) 4: Download file by FTP_ROUTINE\$(20, ...) 5: Upload file by FTP_ROUTINE\$(18, ...) 6: Append to file by FTP_ROUTINE\$(19, ...)
		Y		If not zero, it refers to possible causes that result in error.
			Z	The 4 th digit refers to the result. 1: Success 2: Fail to execute a specific FTP task

TASK: CONNECT

TCP_EVENT%				Description
5	1	0	1	Open a connection successfully
5	1	1	2	Failed to connect to host (command connection error)
5	1	2	2	Incorrect username or missing parameter
5	1	3	2	Incorrect password or missing parameter
5	1	4	2	Connection lost

TASK: GET DIRECTORY

TCP_EVENT%				Description
5	2	0	1	Get directory successfully
5	2	1	2	Failed to open local file

5	2	2	2	Failed to open data connection
5	2	3	2	Failed to save data
5	2	4	2	Connection error or lost

TASK: CHANGE DIRECTORY

TCP_EVENT%				Description
5	3	0	1	Change directory successfully
5	3	0	2	Failed to change working directory at host

TASK: UPLOAD FILE

TCP_EVENT%				Description
5	5	0	1	Transmit a file successfully
5	5	1	2	Failed to find local file at terminal (= no file to send)
5	5	2	2	Failed to open data connection
5	5	3	2	Connection error or lost

TASK: APPEND TO FILE

TCP_EVENT%				Description
5	6	0	1	Transmit data and append to a file successfully
5	6	1	2	Failed to find local file at terminal (= no file to send)
5	6	2	2	Failed to open data connection
5	6	3	2	Connection error or lost

TASK: DOWNLOAD FILE

TCP_EVENT%				Description
5	4	0	1	Receive a file successfully
5	4	1	2	Failed to open local file
5	4	2	2	Failed to open data connection
5	4	3	2	Failed to save data
5	4	4	2	Connection error or lost

TASK: RENAME FTP FILES

TCP_EVENT%				Description
5	7	0	1	An FTP file is renamed successfully.

5	7	4	2	Connection error or lost.
5	7	5	2	File doesn't exist.
5	7	6	2	File already exists

TASK: DELETE FTP FILES

TCP_EVENT%				Description
5	8	0	1	An FTP file is deleted successfully.
5	8	4	2	Connection error or lost.
5	8	5	2	File doesn't exist.

Index

#

#fOrMaT • 86
#mOdEm • 87
#SeRIAl • 87
#vErSiOn? • 88

B

BT_INQUIRY\$ • 43
BT_PAIRING • 44

C

CLOSE_COM • 12
COM_DELIMITER • 8

D

DNS_RESOLVER • 17

F

FTP_ROUTINE\$ • 69

G

GET_CTS • 7
GET_NET_PARAMETER\$ • 26, 34, 38, 62
GET_NET_STATUS • 32
GET_TCPIP_MESSAGE • 21
GET_WLAN_STATUS • 34
GSM_CHANGE_PIN • 51
GSM_CHECK_PIN • 51
GSM_SET_PINLOCK • 52

I

IP_CFG or IP_CONFIGURE • 16
IRDA_STATUS • 13
IRDA_TIMEOUT • 8

N

NCLOSE • 17, 65
NREAD\$ • 19
NWRITE • 19

O

OPEN_COM • 12

R

READ_COM\$ • 14

S

SET_COM • 10
SET_COM_TYPE • 9
SET_NET_PARAMETER • 27, 35, 62
SET_RTS • 7
SOCKET_CAN_SEND • 19
SOCKET_HAS_DATA • 20
SOCKET_IP • 16
SOCKET_OPEN • 20
START TCPIP • 31
STOP TCPIP • 31

T

TCP_ERR_CODE • 22
TCP_OPEN • 18, 63

U

UPDATE_BASIC • 75

W

WIFI_SCAN • 38
WRITE_COM • 14