Introduction

This manual introduces communication protocols and programs used for the MICRO^3 and MICRO^{3}C programmable controllers in the 1:1 and 1:N communication computer link systems.

Unless otherwise specified, all functions and descriptions relating the MICRO^3 in this manual apply to both the MICRO^3 and MICRO^{3}C. Where only MICRO^{3}C is applicable, "MICRO^{3}C only" is indicated.

The computer link system makes it possible to control and monitor a maximum of 32 MICRO^3 units connected in a network using a personal computer. Users can create a computer program to send and receive a user program to and from the MICRO^3, start and stop the MICRO^3 operation, change data of data registers, change timer and counter preset values, and collect data from the MICRO^3. Operation status and error data can also be read to the computer. The collected data can be stored and printed out by creating a proper program.

The CUBIQ software is available optionally to edit MICRO^3 user programs and monitor MICRO^3 operation in the 1:1 and 1:N communication computer link systems.

Features

- A maximum of 32 MICRO^3 units can be controlled and monitored from a computer.
- MICRO^3 units can be connected to the 1:N communication computer link system using a shielded 2-core twisted pair cable. The total length of the cable can be 200 meters (656 feet) at the maximum.
- Data can be read from MICRO^3 units easily by sending an appropriate communication message from a computer.
- The 1:1 communication computer link system can be set up simply by connecting MICRO^3 to a computer using computer link cable FC2A-KC2, which is 2 meters (6.56 feet) long. For the MICRO^{3}C 1:1 computer link, see page 1-3.

Functions

- **Write data from computer to MICRO^3**

  User program
  Inputs, outputs, internal relays, and shift registers in N bytes or 1 bit
  Timer/counter preset values, data registers, and calendar/clock

- **Read data from MICRO^3 to computer**

  User program
  Inputs, outputs, internal relays, and shift registers in N bytes or 1 bit
  Timer/counter preset and current values, data registers, and calendar/clock
  High-speed counter preset and current values
  Error code
  PLC operating status, timer/counter preset value change, user program protection, and MICRO^3 base unit type
  Scan time
  PLC system program version
  User communication transmit/receive buffer (MICRO^{3}C only)
  User communication status (MICRO^{3}C only)
  Communication mode (MICRO^{3}C only)

- **Clear operand data**

  Inputs, outputs, internal relays, timer/counter preset value changed data, timer/counter current values, and data register
  All of inputs, outputs, internal relays, timer/counter current values, and data registers
  Error code
  Link formatting sequence
  User communication data to start user communication data monitor (MICRO^{3}C only)

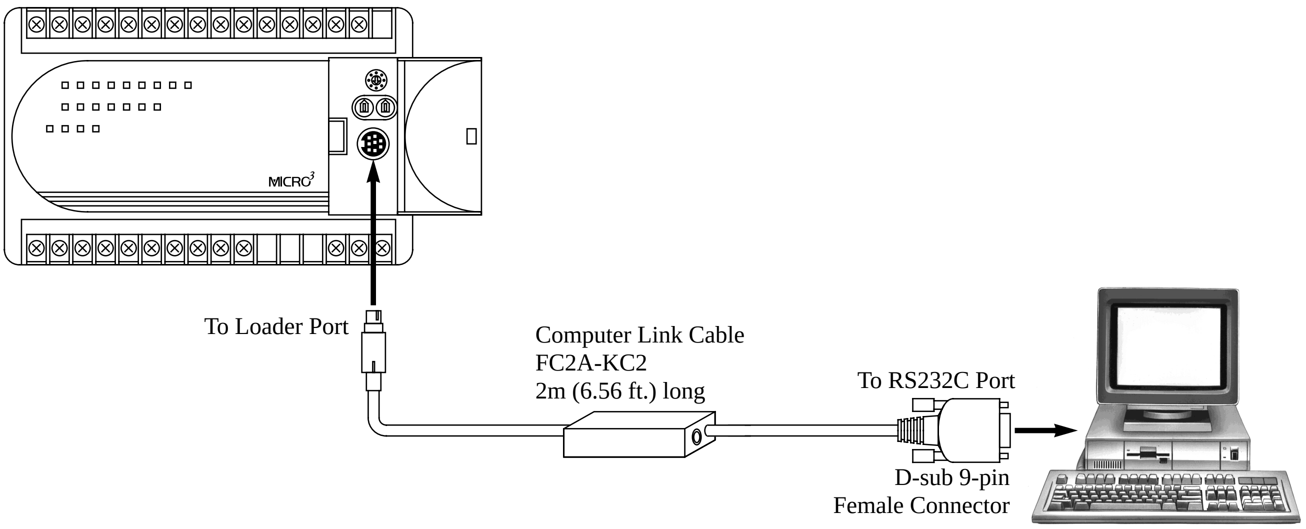
Requirements

To create a computer program for the MICRO^3 computer link system, prepare the following tools:

- Programming language for computer, such as BASIC
- Manual supplied with the computer
- Manual for the OS such as MS-DOS (MS-DOS is a trademark of Microsoft Corporation.)
- A protocol analyzer is recommended to check communication data.

Computer Link 1:1 Communication (MICRO^3)

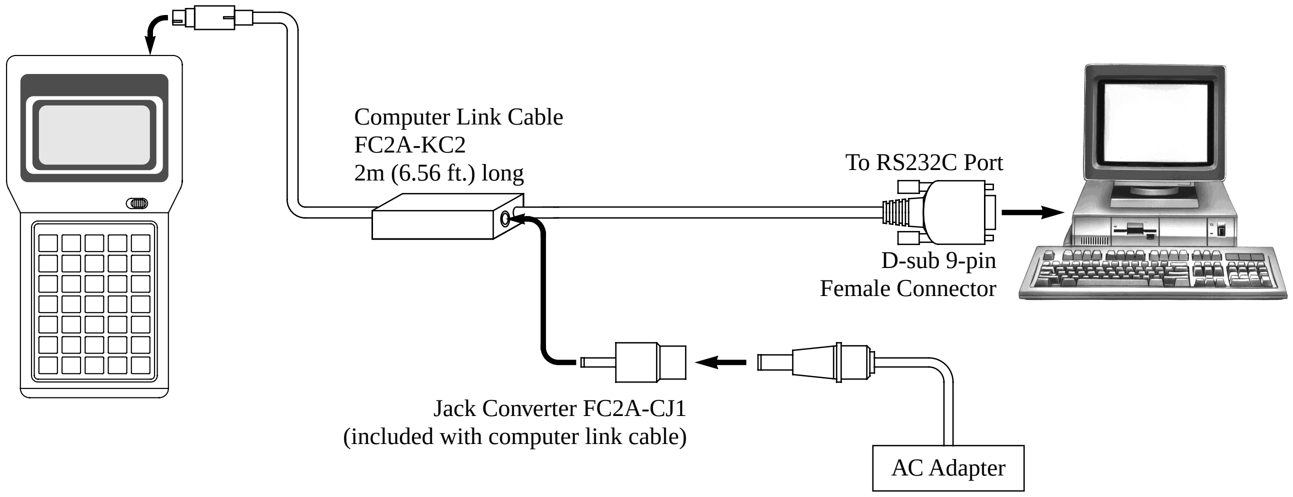
The 1:1 communication computer link system is set up using MICRO^3 and an IBM PC or compatible computer connected with the computer link cable FC2A-KC2, 2m (6.56 ft.) long. When a longer distance is needed, the cable can be extended up to 200m (656 ft.) using the 1:N communication computer link system.



Use FUN8 loader port communication mode setting to make sure that the communication parameters for the MICRO^3 loader port are the same as the computer connected. For FUN8, see MICRO^3 user's manual EM317.

Communication between the program loader and computer

The program loader can also be connected to an IBM PC or compatible using computer link cable FC2A-KC2 for communication. An AC adapter is required to power the program loader. Connect the computer link cable to the loader cable connection port on the program loader. Plug the jack converter into the converter box on the computer link cable, and plug the AC adapter into the jack converter.

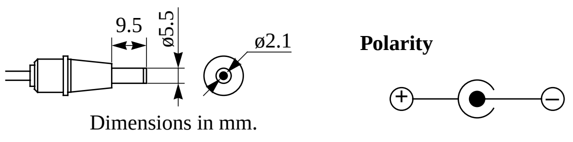


AC Adapter

When using the program loader for communication with a computer, an AC adapter is required to power the program loader. AC adapter output capacity: 5 to 6.5V DC, 4W

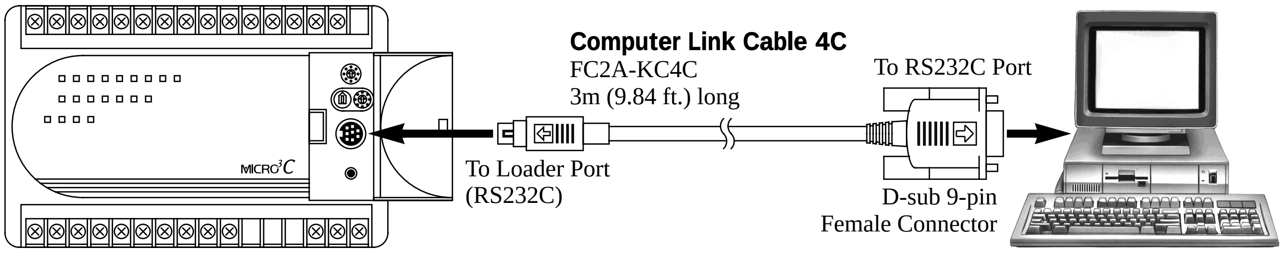
The RS232C/RS485 converter is powered by 24V DC source or an AC adapter with 9V DC, 350mA output capacity.

The output plug of the AC adapter applicable to both the program loader and RS232C/RS485 converter is shown on the right.



Computer Link 1:1 Communication through Loader Port (MICRO^{3}C)

To set up a 1:1 computer link system, connect an IBM PC or compatible to the MICRO^{3}C using the computer link cable 4C (FC2A-KC4C). Set the protocol selector switch to 0, 2, or 4 to select loader protocol for the loader port.

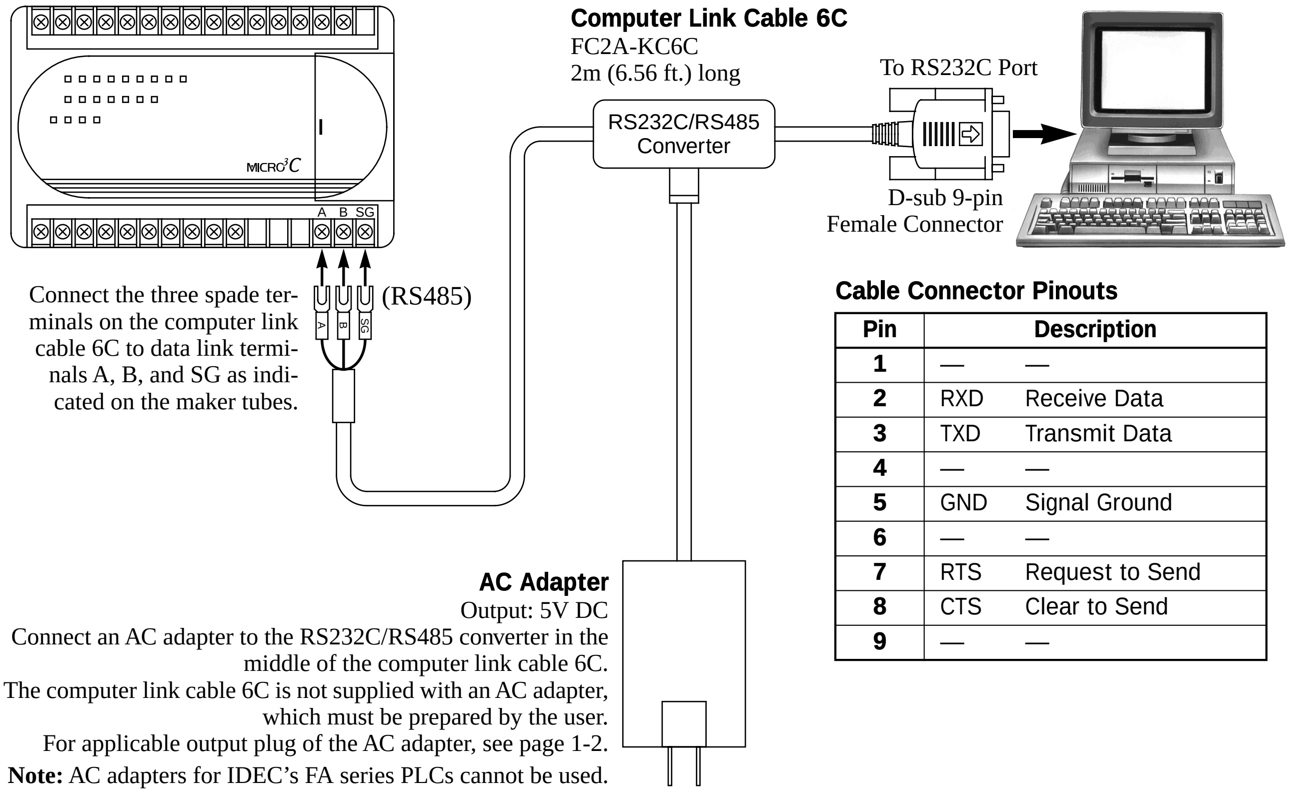


Cable Connector Pinouts

Pin	Description	
1	DCD	Data Carrier Detect
2	RXD	Receive Data
3	TXD	Transmit Data
4	DTR	Data Terminal Ready
5	GND	Signal Ground
6	DSR	Data Set Ready
7	—	—
8	CTS	Clear to Send
9	—	—

Computer Link 1:1 Communication through Data Link Terminals (MICRO^{3}C)

A 1:1 computer link system can also be set up through the data link terminals on the MICRO^{3}C using the computer link cable 6C (FC2A-KC6C). Set the protocol selector switch to 2, 3, or 4 to select loader protocol for the data link terminals.



Connect the three spade terminals on the computer link cable 6C to data link terminals A, B, and SG as indicated on the maker tubes.

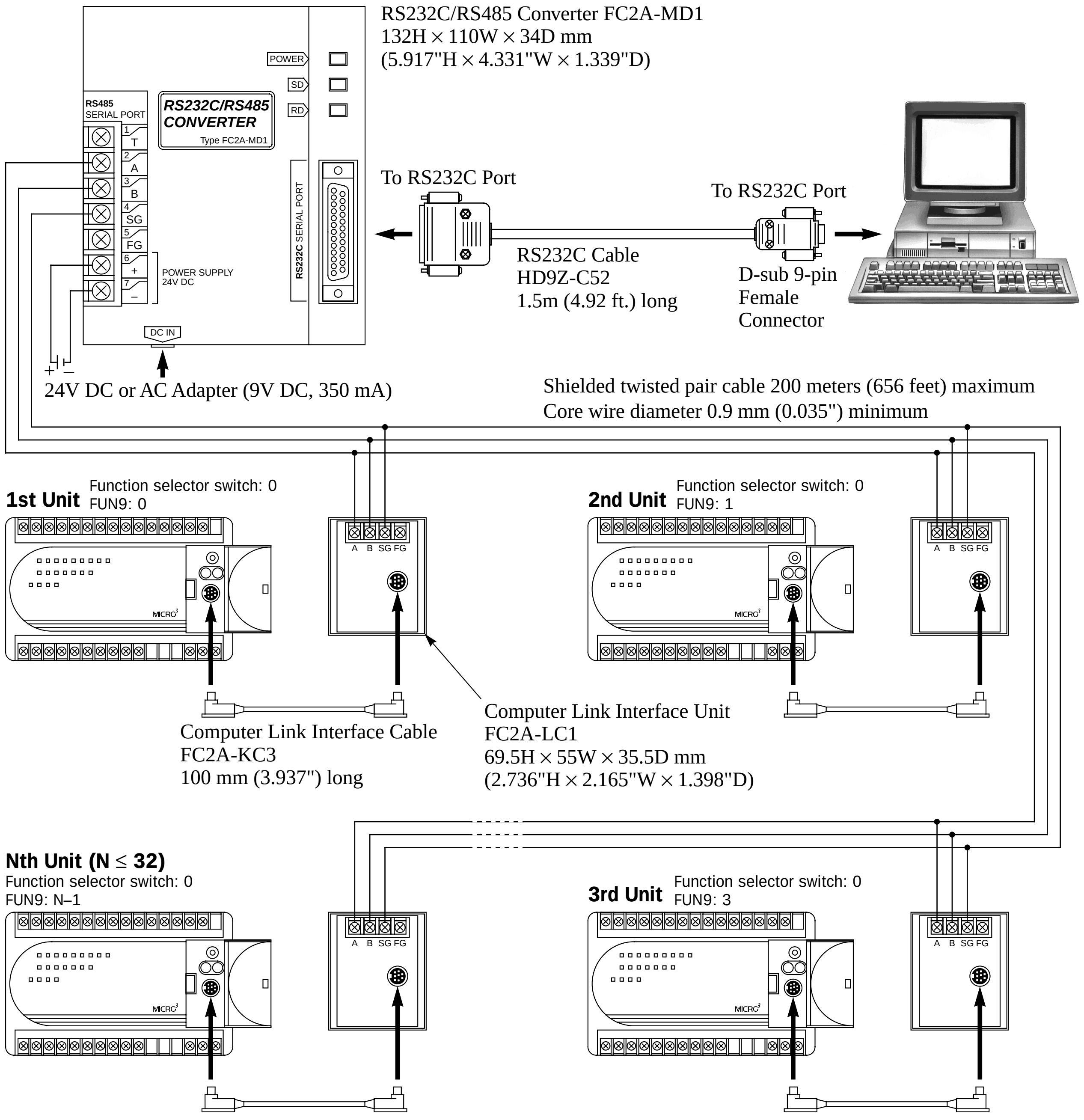
AC Adapter
Output: 5V DC
Connect an AC adapter to the RS232C/RS485 converter in the middle of the computer link cable 6C.
The computer link cable 6C is not supplied with an AC adapter, which must be prepared by the user.
For applicable output plug of the AC adapter, see page 1-2.
Note: AC adapters for IDEC's FA series PLCs cannot be used.

Cable Connector Pinouts

Pin	Description	
1	—	—
2	RXD	Receive Data
3	TXD	Transmit Data
4	—	—
5	GND	Signal Ground
6	—	—
7	RTS	Request to Send
8	CTS	Clear to Send
9	—	—

Computer Link 1:N Communication (MICRO3)

To set up a 1:N computer link system, connect a computer to RS232/RS485 converter using RS232C cable HD9Z-C52. Connect the RS232C/RS485 converter to computer link interface units FC2A-LC1 using shielded twisted pair cables. Connect MICRO3 to each computer link interface unit using computer link interface cable FC2A-KC3.
Supply power to the RS232C/RS485 converter by connecting a 24V DC source to terminals 6 and 7 or by plugging an AC adapter to the DC IN jack. For specifications of the AC adapter, see page 1-2.


RS232C/RS485 Converter FC2A-MD1
132H × 110W × 34D mm
(5.917"H × 4.331"W × 1.339"D)
POWER
SD
RD
RS485 SERIAL PORT
RS232C/RS485 CONVERTER
Type FC2A-MD1
1 T
2 A
3 B
4 SG
5 FG
6 +
7 –
POWER SUPPLY 24V DC
RS232C SERIAL PORT
DC IN
To RS232C Port
To RS232C Port
RS232C Cable
HD9Z-C52
1.5m (4.92 ft.) long
D-sub 9-pin Female Connector
24V DC or AC Adapter (9V DC, 350 mA)
Shielded twisted pair cable 200 meters (656 feet) maximum
Core wire diameter 0.9 mm (0.035") minimum
1st Unit
Function selector switch: 0
FUN9: 0
2nd Unit
Function selector switch: 0
FUN9: 1
A B SG FG
Computer Link Interface Cable
FC2A-KC3
100 mm (3.937") long
Computer Link Interface Unit
FC2A-LC1
69.5H × 55W × 35.5D mm
(2.736"H × 2.165"W × 1.398"D)
Nth Unit (N ≤ 32)
Function selector switch: 0
FUN9: N–1
3rd Unit
Function selector switch: 0
FUN9: 3

In place of the computer link interface cable FC2A-KC3, loader cable FC2A-KL1 (2m/6.56 ft. long) or FC2A-KL2 (5m/16.4 ft. long) can also be used to connect MICRO3 to the computer link interface unit.

Use FUN8 loader port communication mode setting to make sure that the communication parameters for the MICRO3 loader port are the same as the computer connected.

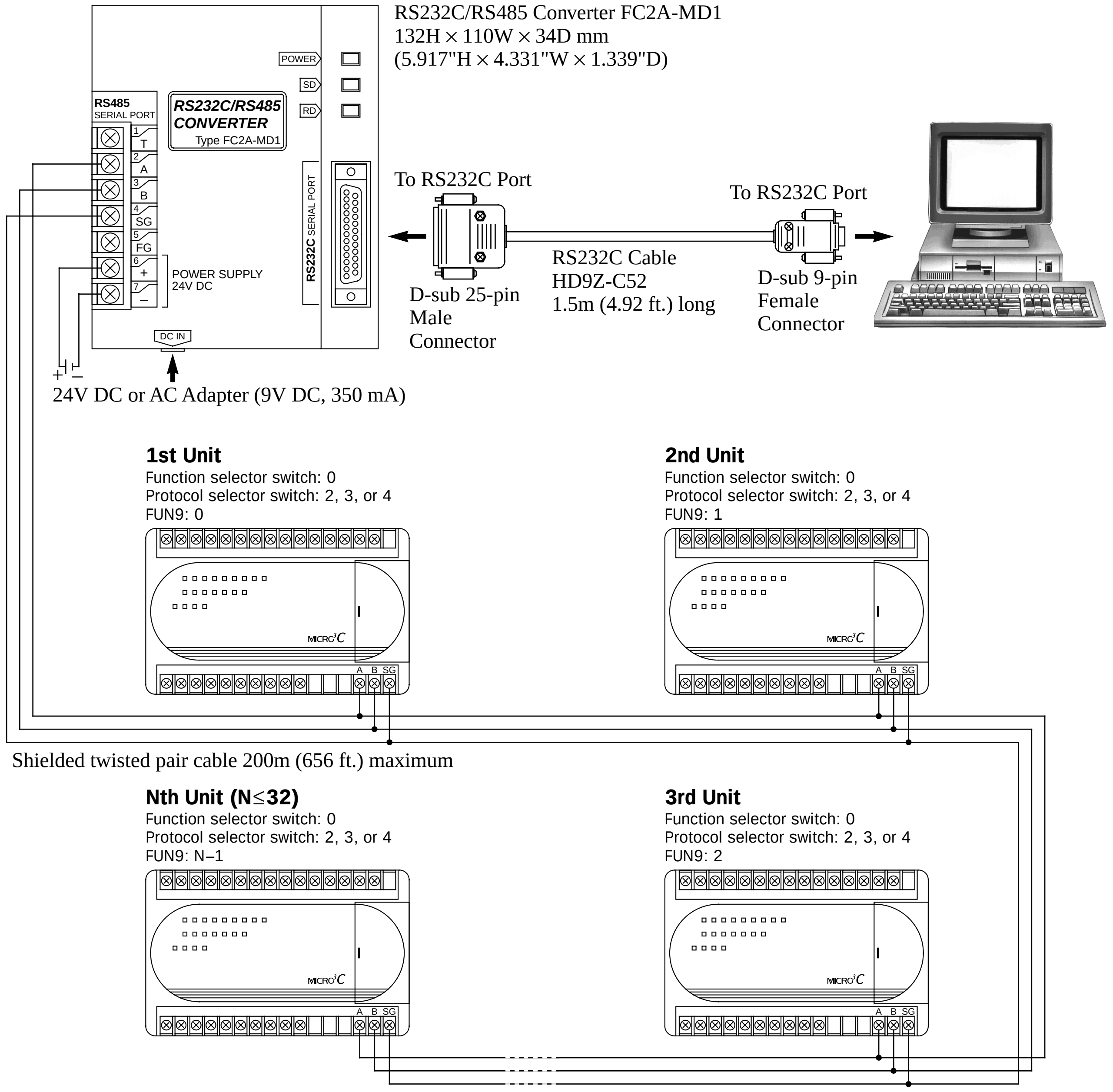
Select a unique device number, from 0 through 31, for each MICRO3 using FUN9 PLC address for network communication on the program loader, and transfer the user program to the MICRO3.

Computer Link 1:N Communication (MICRO^{3}C)

Unlike the computer link 1:N communication system for the MICRO^3, shielded twisted pair cables from the RS232C/RS485 converter can be connected to data link terminals on the MICRO^{3}C directly, without the need for the computer link interface units and computer link interface cables.

To set up a 1:N computer link system, connect a computer to RS232C/RS485 converter using RS232C cable HD9Z-C52. Connect the RS232C/RS485 converter to MICRO^{3}C units using shielded twisted pair cables.

Supply power to the RS232C/RS485 converter by connecting a 24V DC source to terminals 6 and 7 or by plugging an AC adapter to the DC IN jack. For specifications of the AC adapter, see page 1-2.



RS232C/RS485 Converter FC2A-MD1
132H × 110W × 34D mm
(5.917"H × 4.331"W × 1.339"D)
POWER
SD
RD
RS485 SERIAL PORT
RS232C/RS485 CONVERTER
Type FC2A-MD1
1 T
2 A
3 B
4 SG
5 FG
6 +
7 –
POWER SUPPLY 24V DC
RS232C SERIAL PORT
DC IN
24V DC or AC Adapter (9V DC, 350 mA)
To RS232C Port
To RS232C Port
RS232C Cable HD9Z-C52 1.5m (4.92 ft.) long
D-sub 25-pin Male Connector
D-sub 9-pin Female Connector
1st Unit
Function selector switch: 0
Protocol selector switch: 2, 3, or 4
FUN9: 0
2nd Unit
Function selector switch: 0
Protocol selector switch: 2, 3, or 4
FUN9: 1
Shielded twisted pair cable 200m (656 ft.) maximum
Nth Unit (N≤32)
Function selector switch: 0
Protocol selector switch: 2, 3, or 4
FUN9: N–1
3rd Unit
Function selector switch: 0
Protocol selector switch: 2, 3, or 4
FUN9: 2
A B SG
MICRO3C

Select a unique device number, from 0 through 31, for each MICRO^{3}C using FUN9 PLC address for network communication on the program loader, and transfer the user program to the MICRO^{3}C.

Loader Port Communication Specifications (MICRO^3)

The MICRO^3 base unit and the program loader have RS485 interface to communicate with each other in the RS485 signal level. To communicate with a computer, the RS485 signals must be converted into RS232C signals.

Electrical Characteristics		Compliance with EIA standard RS485	
Communication Method		Half-duplex	
Synchronization		Start-stop synchronization	
Communication Configuration		1:N	
Communication Format	Baud Rate	1200, 2400, 4800, 9600, 19200 bps (Default: 9600 bps)	
	Data Bits	Start bit	1
		Data bit	7, 8 (Default: 7)
		Parity bit	Even, Odd, None (Default: Even)
		Stop bit	1, 2 (Default: 1)
DTR/DSR Control		Available (when using the RS232C/RS485 converter)	
Maximum Cable Length		200m (656 ft.) total using twisted pair cable	
Receive Timeout		10 to 2550 msec (10 msec increments) designated using FUN8 MICRO^3 base unit receive timeout default value 500 msec	
Communication Device Number		0 through 31 (Default: 0) 255 (used by the program loader to access all device numbers)	

Program Loader Communication Format

The program loader uses a fixed communication format of baud rate and data bits which are the default values of the MICRO^3 base unit.

Communication Format	Baud Rate	9600 bps	
	Data Bits	Start bit	1
		Data bit	7
		Parity bit	Even
		Stop bit	1

Loader Port Communication Specifications (MICRO^{3}C)

The loader port on the MICRO^{3}C has RS232C interface to communicate with a computer and other RS232C equipment directly.

Standards		EIA RS232C
Maximum Cable Length		15m (49.2 ft.)
Communication Parameters	**Baud Rate**	1200, 2400, 4800, 9600, 19200 bps
	Data Bits	7 or 8 bits
	Parity	Odd, Even, None
	Stop Bits	1 or 2 bits
	Receive Timeout	10 to 2550 msec (In the user communication, receive timeout is disabled when 2550 msec is selected.)
Connection to Program Loader		Using optional loader cable 3C (FC2A-KL3C)
Connection to RS232C Equipment		Using optional user communication cable 1C (FC2A-KP1C) or other cables

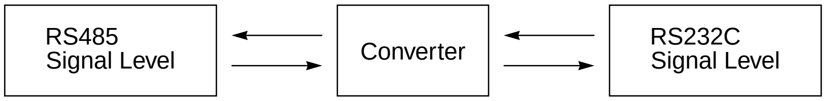
Data Link Terminal Communication Specifications (MICRO^{3}C)

The MICRO^{3}C can communicate with a computer through the data link terminals in a 1:N computer link configuration using the RS232C/RS485 converter.

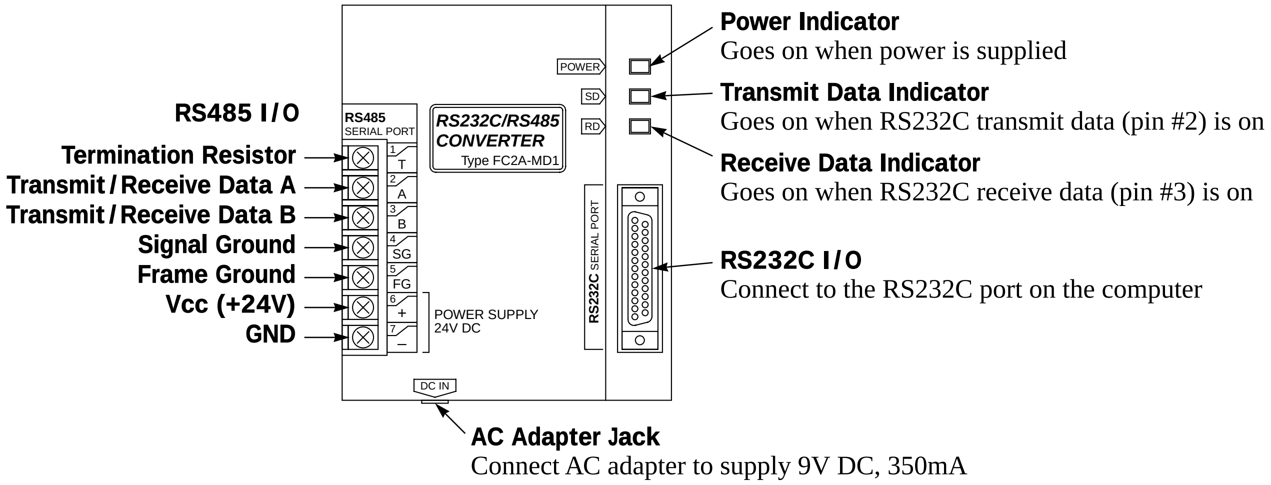
Standards	EIA RS485 (termination resistor is not required)
Recommended Cable	ø0.9 mm shielded twisted cable
Conductor Resistance	85 Ω/km maximum
Shield Resistance	12 Ω/km maximum
Maximum Cable Length	200m (656 ft.)
Isolation	Between data link terminals of multiple MICRO^{3}C units: Not isolated
Baud Rate	Expansion or data link communication: 19200 bps (fixed) Loader protocol communication: 9600 bps (fixed)
Communication Delay	Expansion link: Master station normal scan time + approx. 9 to 10 msec Data link: Master station normal scan time + approx. 12.5 to 13 msec + Slave station scan time
Connection to Program Loader	Using optional loader cable 4C (FC2A-KL4C)

RS232C / RS485 Converter FC2A-MD1

The RS232C/RS485 converter FC2A-MD1 is used with the MICRO^{3}C and the MICRO^3 to convert data signals between EIA RS232C and EIA RS485. This converter makes it possible to connect a host device with RS232C interface to multiple MICRO^{3}C and MICRO^3 programmable controllers using one cable.



Parts Description



Note: The FC2A-MD1 contains a 220Ω termination resistor on the RS485 line, eliminating the need for an external termination resistor. To use the internal termination resistor, connect terminal T to terminal B. When the termination resistor is not needed, disconnect terminal T from terminal B.

General Specifications

Rated Power Voltage	Power terminals DC IN adapter jack	24V DC ±20% (Ripple 10% maximum) 9V DC, 350mA supplied from AC adapter
Current Draw	Power terminals: Approx. 40 mA at the rated voltage	
Operating Temperature	0 to 60°C	
Storage Temperature	–20 to +70°C	
Operating Humidity	45 to 85% RH (no condensation)	
Vibration Resistance	5 to 55 Hz, 60 m/sec^2, 2 hours each in 3 axes	
Shock Resistance	300 m/sec^2, 3 shocks each in 3 axes	
Dielectric Strength	1500V AC, 1 minute between live parts and dead parts	
Insulation Resistance	10 MΩ minimum between live parts and dead parts (500V DC megger)	
Noise Resistance	Power terminals: ±1 kV, 1 µsec (using noise simulator)	
Weight	Approx. 550g	

Serial Interface Specifications

Standards in Compliance	EIA standard RS232C (D-sub 25-pin female connector) EIA standard RS485 (screw terminals)
Communication Method	Half-duplex
Communication Configuration	1:N (N ≤ 32)
Communication Cable	Shielded twisted-pair cable
Communication Baud Rate	9600 bps (fixed)
Slave Stations	32 slave stations maximum (RS485 line)
Maximum Cable Length	RS232C: 15m (49.2 ft.) RS485: Total 200m (656 ft.)

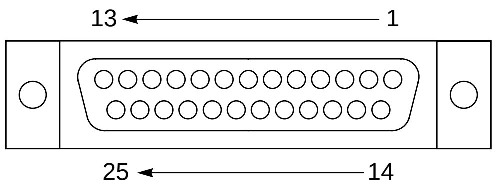
RS232C / RS485 Converter FC2A-MD1, continued

RS485 Terminal Arrangement

Terminal No.	Symbol	Name
1	SG	Signal Ground
2	SD A	Transmit Data A
3	SD B	Transmit Data B
4	FG	Frame Ground
5	SG	Signal Ground
6	RD A	Receive Data A
7	RD B	Receive Data B
8	FG	Frame Ground
9	+	Vcc (+24V)
10	–	GND

RS232C Connector Pin Arrangement

Pin No.	Symbol	Name
1	GND	Frame Ground
2	TXD	Transmit Data
3	RXD	Receive Data
4	RTS	Request to Send
5	CTS	Clear to Send
6	(NC)	Unused
7	GND	Signal Ground
8-25	(NC)	Unused



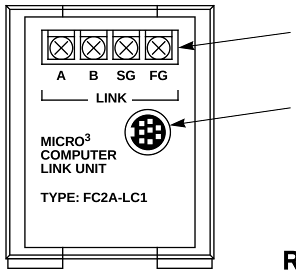
25-pin Female Connector on RS232C/RS485 Converter

Computer Link Interface Unit FC2A-LC1

One computer link interface unit is used with each MICRO^3 unit in the 1:N communication computer link system.

Note: MICRO^{3}C does not require the computer link interface unit to set up the 1:N communication computer link system.

The computer link interface unit cannot be connected to the program loader.



RS-485 Terminals
Connect to the RS232C/RS485 converter.

Cable Connector
Connect the computer link interface cable FC2A-KC3 to MICRO^3.

For dimensions of the computer link interface unit, see page 14-2.

RS485 Terminal Arrangement

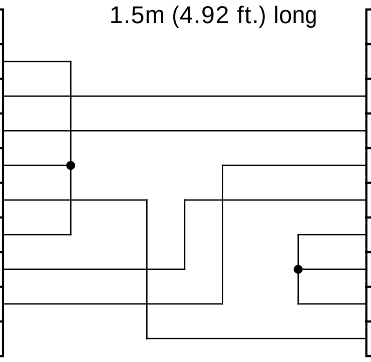
Symbol	Name
A	Transmit/Receive Data A
B	Transmit/Receive Data B
SG	Signal Ground
FG	Frame Ground

RS232C Cable HD9Z-C52

Connector for Computer

Symbol	Pin No.
DCD	1
RXD	2
TXD	3
DTR	4
GND	5
DSR	6
RTS	7
CTS	8
RI	9

D-sub 9-pin female connector



Connector for RS232C / RS485 Converter

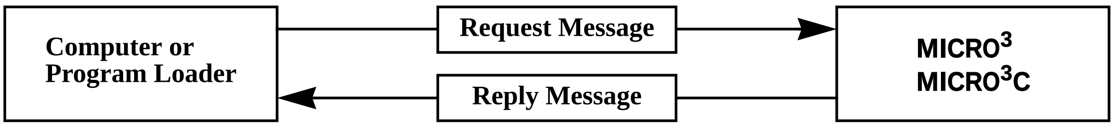
Pin No.	Symbol	Name
1	GND	Frame Ground
2	TXD	Transmit Data
3	RXD	Receive Data
4	RTS	Request to Send
5	CTS	Clear to Send
6	DSR	Data Set Ready
8	DCD	Data Carrier Detect
20	DTR	Data Terminal Ready
7	GND	Signal Ground

D-sub 25-pin male connector

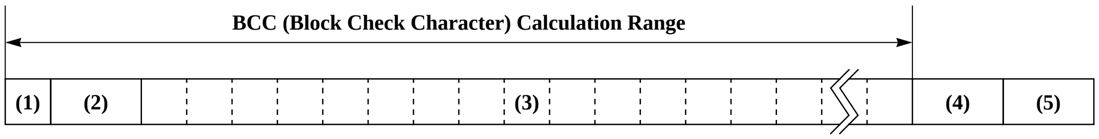
Communication Procedure

The computer and MICRO^3/MICRO^{3}C base unit communicate data by sending and receiving communication messages, which consist of request messages and reply messages. The request message is sent from the computer to write data to or read data from MICRO^3. The reply message is sent from MICRO^3 in response to the request message from the computer.

Communication is always initiated by the computer by sending a request message to MICRO^3, which then returns a reply message to the computer. MICRO^3 cannot initiate communication.



Message Format



Communi-cation Message					
(1)	Communication control character (1 byte)	Message start character	ENQ (05h)	Enquiry	Request message
			ACK (06h) NAK (15h)	Acknowledge Negative acknowledge	Reply message
(2)	Communication device number (2 bytes)	Device number to send request to	00 (0) through 1F (31)	Designates MICRO^3 device number (FUN9) to which the computer sends a request message in the 1:N communication computer link system.	
			FF (255)	Used in the 1:1 communication computer link system. MICRO^3 of any device number receives request message.	
		Device number to send reply from	00 (0) through 1F (31)	Indicates the device number (FUN9) of MICRO^3 which returns the reply message.	
(3)	Data (variable length)	Communication command, data type, etc.	Depends on each command. See "Request Messages" on page 3-2. See "Reply Messages" on page 3-4.		
(4)	BCC (2 bytes)	Block check character	Exclusive OR (XOR) of the BCC calculation range.		
(5)	Terminator (1 or 2 bytes)	Message end code	CR (0Dh)	Default	
			CR (0Dh) + LF (0Ah)	Selected using FUN8 (loader port communication mode setting)	

Request Messages

Request messages are available in request message 1 and request message 2 with different data structures.

Request Message 1

Request message 1 is a command message to be sent from the computer to MICRO^3, containing a command. The data type code included in the request message determines the function. The data structure of request message 1 is shown below:

	ENQ							
Request Message 1	05h	Device	(1)	(2)	(3)	(4)	BCC	Terminator

(1)	Continuation (1 byte)	0 (30h)	Discontinued (no message follows)				
		1 (31h)	Continued (another message follows)				
(2)	Command (1 byte)	W (57h)	Write data to MICRO^3				
		R (52h)	Read data from MICRO^3				
		C (43h)	Clear data from MICRO^3				
(3)	Data type (1 byte)	X (58h)	Input	N-byte designation	t (74h)	Timer (current value)	N-byte designation
		Y (59h)	Output				
		M (4Dh)	Internal relay		c (63h)	Counter	
		R (52h)	Shift register		x (78h)	Input	1-bit designation
		T (54h)	Timer (preset value)		y (79h)	Output	
		C (43h)	Counter (preset value)		m (6Dh)	Internal relay	
		D (44h)	Data register		r (72h)	Shift register	
		U (55h)	High-speed counter (preset value + current value)				
		P (50h)	User program				
		S (53h)	PLC operating status				
		N (4Eh)	PLC system program version				
		K (4Bh)	Scan time				
		W (57h)	Calendar/clock				
		E (45h)	Error code				
		Z (5Ah)	System reset				
		I (49h)	Link formatting sequence				
		G (47h)	User communication receive buffer	MICRO^{3}C only			
		g (67h)	User communication transmit buffer	MICRO^{3}C only			
		A (41h)	19,200 bps (clear data)	MICRO^{3}C only			
		B (42h)	9,600 bps (clear data) User communication status (read data)	MICRO^{3}C only MICRO^{3}C only			
		H (48h)	Communication mode (read data)	MICRO^{3}C only			
(4)	Data (variable length)	Data (depends on command and data type)					

(1) "Continued" is used in request message 1 for writing the user program to inform MICRO^3 that another request message will be sent successively. In all other request messages, "discontinued" is used. When "continued" is specified, the computer sends a request message, receives a reply message, and sends another request message.

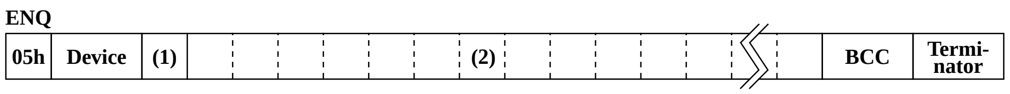
(2) The command code is available in three types; write data, read data, and clear data.

(3) The data type code selects an operand or function. Upper- and lower-case characters have different functions.

(4) The data specifies the operand number, the quantity of bytes of the data for reading or writing, etc. depending on the command and data type.

Request Message 2

Request message 2 is a command message used for writing and reading user programs. The data structure of request message 2 is shown below:

Request Message 2			
(1)	Continuation (1 byte)	0 (30h)	Discontinued (no message follows)
(2)	Data (variable length)	User program (write user program)	
	Data (1 byte)	R (52h)	Read user program

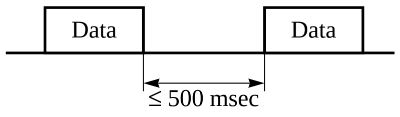
(1) "Discontinued" is used for both writing and reading user programs to inform MICRO3 that no request message will be sent successively.

(2) The data length is variable for writing user programs and is 1-byte long ("R") for reading user programs.

Receive Timeout

When a request message contains an interval of 500 msec or more between one-byte character data and the next one-byte character data, MICRO3 understands that the communication is canceled and does not return a reply message.

When the interval is 500 msec or more, extend the receive timeout value using FUN8 (loader port communication mode setting). The receive timeout can be selected between 10 and 2550 msec in 10-msec increments. To enable the optional communication mode, turn on the mode selection input designated by FUN8.



Reply Messages

Reply messages are available in ACK reply message and NAK reply message with different data structures.

ACK Reply Message

The ACK reply message is a reply or response to the request message and is sent from MICRO3 to the computer when communication is completed normally.

ACK Reply Message	ACK: 06h | Device | (1) | … | (2) | … | BCC | Terminator				
(1)	Command (1 byte)	0 (30h)	OK: Discontinued	All communication is completed normally (end of processing).	
		1 (31h)	OK: Continued	Communication in reply to request is completed normally and another reply message follows when reading a user program.	
		2 (32h)	NG: Error	Communication device number, command, data type, data, or continuation code is not within the range supported by MICRO3 or does not match its status. When this error occurs, communication is halted without regard to the continuation code.	
(2)	Data (variable length)	0 (30h) to 9 (39h) or A (41h) to F (46h)	OK reply	When request command is W or C	No data exists. (0 byte)
				When request command is R	The data length depends on the request command (variable length).
			NG reply	NG code (2 bytes)	

NG code (2 bytes)

NG Code	Error	Cause
00	Expansion station error	Communication attempted to expansion station
01	Program size error	Improper write/read program size
02	Protect error	Protected against write/read in MICRO3
03	RUN error	Writing user program attempted while MICRO3 is running
04	CRC error	User program CRC code not matched
06	Data range error	Invalid data range designated
07	Timer/counter preset value change error	Preset value change attempted to timer or counter with preset value designated by data register
08	Calendar/clock data error	Invalid value written to calendar/clock
09	Data clear error	Designated data cannot be cleared
10	Data error	Invalid data other than 0 (30h) - 9 (39h) or A (41h) - F (46h)
11	Setting error	Incorrect setting for user communication (MICRO3C only)

(1) The command code indicates whether the request command is completed normally or not and also whether another reply message will be sent successively.

When reading a user program from MICRO3, reply message 1 is returned in response to request message 1 and reply message 2 is returned in response to request message 2. Reply message 1 contains command 1 (OK: continued) to inform the computer that another reply message follows. All other reply messages contain command 0 (OK: discontinued) to indicate that no reply message follows when communication is completed normally.

(2) When an OK reply is returned in response to request command R (read data), the read data is included in this place. When an NG reply is returned, the cause of error exists in MICRO3. See page 13-2.

NAK Reply Message

When an error is found during communication, a NAK reply message is sent from MICRO^3 to the computer.

NAK Reply Message

NAK

15h	Device	(1)	(2)	BCC	Termi-nator

(1)	Command	0 (no meaning): dummy data for consistent communication format
(2)	Communication error code (2 bytes)	Depending on the communication error, an error code is set in this place.

Error Code	Error Type	Error Contents
00	BCC error	Appended BCC code does not match BCC calculated value of received data.
01	Frame error	Quantity of received bits differs from the preset value (stop bit is 0 for example).
02	Data send/receive error	Parity error or overrun error occurred.
03	Command error	Unsupported request message is received.
04	Procedure/data quantity error	Received request message does not match the expected data (including quantity of data).

(1) The command code in the NAK reply message is always 0.

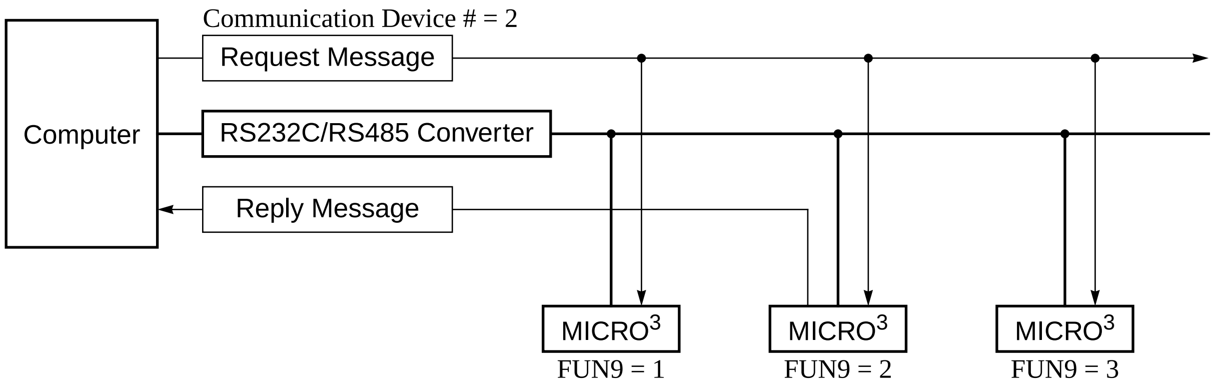
(2) The next two bytes indicate the communication error code.

Communication Device Number in Communication Message

The communication device number is an address number 0 through 31 of the MICRO^3 base unit in a 1:N communication computer link network. The device number is stored in the FUN9 area of the user program in the MICRO^3 base unit. The computer uses the device number to differentiate various MICRO^3 base units that it communicates with. When the communication device number in the request message matches the value stored in FUN9, MICRO^3 returns a reply message.

Communication Procedure Schematic

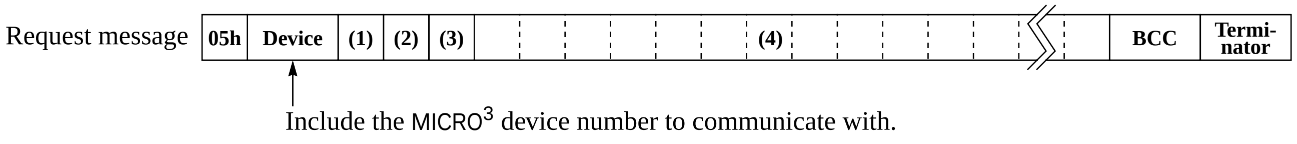
For example, when the computer sends a request message including communication device number 2, MICRO^3 of device number 2 returns a reply message.



A communication device number must also be included in the request message in the 1:1 computer link system. If communication device number 255 is included in the request message, MICRO^3 receives the request message regardless of the FUN9 value and returns a reply message.

Since the program loader sends a request message including device number 255, the program loader can communicate with MICRO^3 base units of any device number and can change the device number of MICRO^3 base units.

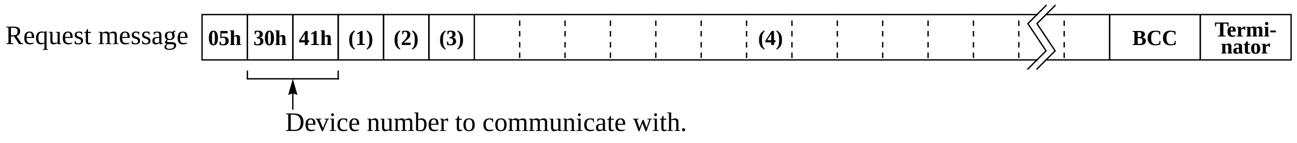
To select a MICRO^3 base unit in a computer link network, include the desired communication device number in the request message.



Example: To specify communication device number 10

Convert decimal value 10 into hexadecimal value 0Ah. Convert each character of the hexadecimal value into hexadecimal ASCII codes of 30h and 41h. Include the two-byte code in place of the device number in the request message.

10 (0Ah) → ASCII codes (30h 41h)



Note: The device number of MICRO^3 is selected using FUN9 on the program loader. After changing the FUN9 value, transfer the user program from the program loader to MICRO^3.

All MICRO^3 base units in a computer link network must have a unique device number 0 through 31. Make sure that the same device number does not exist in a computer link network.

If the device number included in the request message is not found in the computer link network, no response is returned from any MICRO^3 base unit to the computer.

The device number of MICRO^3 can also be changed using the CUBIQ software. Change the FUN9 communication device # in the FUN table. In the transfer menu box, select device number 255, and transfer the user program from the computer to MICRO^3 using the 1:1 communication computer link system.

Communication Processing Time

When monitoring the MICRO^3 status in a computer link system, the communication processing time between the computer and MICRO^3 is required in addition to the processing times at the computer and MICRO^3. When setting up a computer link system, the communication processing time must be taken into consideration.

Calculating the Communication Processing Time

1. From the communication format, calculate the quantity of data in bytes that can be sent and received per second:

   Bytes of data per second = Baud rate (bps) ÷ Communication bit count

   Communication bit count = Start bit + Data bits + Parity bit + Stop bit

   (Parity bit: None = 0, Even or Odd = 1)

2. Calculate the byte count to be communicated.

   Communication byte count = Byte count to send + Byte count to receive

3. Calculate the communication processing time using these values.

   Communication processing time = Communication byte count ÷ Bytes of data per second

Example: Read data register 1-word (2 bytes) data from 10 MICRO^3 base units with the default communication format.

1. The default communication format values are:

Baud rate	9600 bps
Start bit	1 bit
Data bits	7 bits
Parity bit	Even (1 bit)
Stop bit	1 bit

   Therefore, the communication bit count is 10 bits (= 1 + 7 + 1 + 1).
   The characters (bytes) of data communicated per second are 960 bytes (= 9600 ÷ 10).

2. The byte count of the request message is:

Communication control character	1 byte
Communication device number	2 bytes
Continuation code	1 byte
Data	8 bytes
BCC	2 bytes
Terminator	1 byte
Total	15 bytes

The byte count of the reply message is:

Communication control character	1 byte
Communication device number	2 bytes
Command	1 byte
Data	4 bytes *
BCC	2 bytes
Terminator	1 byte
Total	11 bytes

   Therefore, the communication byte count is 26 bytes (= 15 + 11).

   * Although reading 2 bytes of data is specified in the request message, the data of a data register consists of 4 characters and 4 bytes of data is returned in the reply message.

3. The communication processing time for one MICRO^3 base unit is 0.027 sec (= 26 ÷ 960).
   Since the communication is executed for 10 MICRO^3 base units, the total communication processing time will be 0.27 sec (= 0.027 × 10).

For calculating the byte counts of the request and reply messages, see chapter 4.

Selecting MICRO^3 Communication Format

The communication format for MICRO^3 can be changed using FUN8 (Loader Port Communication Mode Setting). This function makes it possible to communicate with a device which has an RS232C interface with fixed communication parameters such as baud rate and terminator code.

The available communication parameters and default values selected with FUN8 are listed below:

Communication Parameter	Option	Default (Standard Mode)
Baud Rate	1200, 2400, 4800, 9600, 19200 bps	9600 bps
Terminator Code	0D (CR), 0D 0A (CR LF)	0D (CR)
Data Bits	7, 8 bits	7 bits
Parity Check	None, Even, Odd	Even
Stop Bits	1, 2 bits	1 bit
Mode Selection Input	I0 to I15	None
Receive Timeout	10 to 2550 (10-msec increments)	500 msec

For setting FUN8 using the program loader, see MICRO^3 User's Manual EM317. For changing FUN table settings using the CUBIQ software, see CUBIQ User's Manual EM292.

When the mode selection input selected by FUN8 is turned on, the optional communication mode is enabled. When the mode selection input is off, the default communication mode is enabled. When using the program loader to communicate with MICRO^3, use the standard communication mode of all default values.

After changing the FUN8 communication format, transfer the user program to the MICRO^3 base unit.

Communication Device Number in MICRO^3

The communication device number is an address number 0 through 31 of the MICRO^3 base unit in a computer link network. The device number is stored in the FUN9 area of the user program in the MICRO^3 base unit. The computer uses the device number to differentiate various MICRO^3 base units that it communicates with. When the communication device number in the request message matches the value stored in FUN9, MICRO^3 returns a reply message.

To set a communication device number in MICRO^3, change the FUN9 value using the program loader and transfer the user program from the program loader to MICRO^3. Allocate a unique device number 0 through 31 to each MICRO^3 in a 1:N communication computer link network. If the same communication number is found at two or more MICRO^3 units in a network, a communication error will result.

In a 1:1 communication computer link system, use of communication device number 0 is recommended although any number 0 through 31 is possible.

When the entire user program is deleted using the DEL, END, ↵ keys on the program loader, the FUN9 is also cleared to the default value of communication device number 0.

For details of setting FUN9, see MICRO^3 User's Manual EM317.

Communication Format for Computer

Set the same communication format for the computer as for MICRO^3. The communication format for the computer is selected by the parameters for opening the communications file. See sample programs shown later in this manual. The default values of the MICRO^3 communication format are even parity, 7 data bits, and 1 stop bit.

Since communication is initiated by sensing a request message from the computer in the MICRO^{3}computer link system, select the start-stop synchronization for the computer using the computer internal clock for timing the sending and receiving operations.

Write User Program

The user program can be written from a computer or program loader to the MICRO^3 base unit.

When writing a user program from a computer, two request messages must be sent to the MICRO^3.
Send request message 1 first. After confirming that the returned reply message is an OK reply, send request message 2.

This function is the same as writing a user program from the program loader by pressing the TRS, ↵, ↵ keys.

Request Messages (Write User Program)

Request Message 1

05h	** **	31h	57h	50h	** ** ** **	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	1 (31h)	Continued
(4)	**Command**	1 byte	W (57h)	Write data
(5)	**Data type**	1 byte	P (50h)	User program
(6)	**Program capacity**	4 bytes	01FA 03FA 07FA	244 steps 500 steps 1K (1012) steps
(7)	**BCC**	2 bytes	00 - 7F	Block check character
(8)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Request Message 2

05h	** **	30h	** ** ** ** ** ** ** … **	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**User program**	Variable length	0 (30h) - 9 (39h) A (41h) - F (46h)	User program (ASCII code file)
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Note: The user program must be stored in a file of the ASCII code format such as a file received from MICRO^3 shown in the sample program on page 7-6. Ladder program files (.LDR) created by the CUBIQ software cannot be sent to MICRO^3 using this request message.

Reply Messages (Write User Program)

OK Reply (Reply to Request Messages 1 and 2)

06h	** **	30h	** **	0Dh
(1)	(2)	(3)	(4)	(5)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**BCC**	2 bytes	00 - 7F	Block check character
(5)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

NG Reply (Reply to Request Message 1)

06h	** **	32h	30h 3*h	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	2 (32h)	NG
(4)	**NG code**	2 bytes	01 (30h 31h) 02 (30h 32h) 03 (30h 33h) 04 (30h 34h)	Program capacity error Protect error RUN error CRC error
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Note: NG replies are not returned in response to reply message 2.

Read User Program

The user program can be read from the MICRO^3 base unit to a computer or program loader.

When reading a user program to a computer, two request messages must be sent from the computer to the MICRO^3. Send request message 1 first. After confirming that the returned reply message is an OK reply, send request message 2.

Specify a value larger than the user program capacity selected in the MICRO^3 in place of the program capacity in request message 1. Reserve a buffer larger than the specified value. For details, see the sample program on page 7-6.

This function is the same as reading a user program to the program loader by pressing the TRS, ▼, ↵, keys.

Request Messages (Read User Program)

Request Message 1

05h	** **	30h	52h	50h	** ** ** **	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Dummy (no meaning)
(4)	**Command**	1 byte	R (52h)	Read data
(5)	**Data type**	1 byte	P (50h)	User program
(6)	**Program capacity**	4 bytes	0000 - FFFF	User program receive buffer size
(7)	**BCC**	2 bytes	00 - 7F	Block check character
(8)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Request Message 2

05h	** **	30h	52h	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Dummy (no meaning)
(4)	**Command**	1 byte	R (52h)	Read data
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Reply Messages (Read User Program)

OK Reply

- **Reply Message 1**

06h	**	**	31h	**	**	**	**	**	**	0Dh
(1)	(2)		(3)		(4)			(5)		(6)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	1 (31h)	OK: Continued
(4)	**Program capacity**	4 bytes	01FA 03FA 07FA	244 steps 500 steps 1K (1012) steps
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

- **Reply Message 2**

06h	**	**	30h	**	**	**	**	**	**	**	**	**	**	0Dh
(1)	(2)		(3)				(4)					(5)		(6)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**User program**	Variable length	0 (30h) - 9 (39h) A (41h) - F (46h)	User program (ASCII code file)
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Note: The received user program is stored on the disk in the ASCII code format.

NG Reply (Reply to Request Message 1)

06h	**	**	32h	30h	3*h	**	**	0Dh
(1)	(2)		(3)	(4)		(5)		(6)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	2 (32h)	NG
(4)	**NG code**	2 bytes	01 (30h 31h) 02 (30h 32h)	Program capacity error Protect error
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Note: NG replies are not returned in response to reply message 2.

Write N Bytes

Data can be written into N-bytes of operands starting with the specified operand number in the MICRO^3 base unit.

This command can be used to turn on or off bit operands such as inputs, outputs, internal relays, and shift register bits in units of 8 bits.

This command can also be used to change timer and counter preset values, enter data into data registers, and set data of calendar and clock (FUN28).

Request Message (Write N Bytes)

05h	** **	30h	57h	**	** ** ** **	** **	** ** ** ** ⟩⟩ **	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**Command**	1 byte	W (57h)	Write data
(5)	**Data type**	1 byte	See table below.	N-byte designation
(6)	**Operand number**	4 bytes	See table below.	First operand number to write to
(7)	**Data length**	2 bytes	00 - C8	Byte count of data to write 200 (C8h) bytes maximum
(8)	**Data**	Variable length	0 (30h) - 9 (39h) A (41h) - F (46h)	Data to write
(9)	**BCC**	2 bytes	00 - 7F	Block check character
(10)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

(5) Data type code		**(6) Operand number**		**Remarks**
		High-speed processing	**Standard processing**	
X (58h) **Y (59h)** **M (4Dh)**	Input Output Internal relay	0000 - 0017 0000 - 0017 0000 - 0047	0000 - 0037 0000 - 0037 0000 - 0287	The least significant digit of the operand number is an octal number (0 through7). Upper digits are decimal numbers.
R (52h) **T (54h)** **C (43h)** **D (44h)** **W (57h)**	Shift register Timer (preset value) Counter (preset value) Data register Calendar/clock	0000 - 0031 0000 - 0015 0000 - 0015 0000 - 0031 0000 - 0006	0000 - 0063 0000 - 0031 0000 - 0031 0000 - 0099* 0000 - 0006	All four digits of the operand number are decimal numbers.

Note*: Data registers can be up to 0099 for the MICRO^3 and up to 0499 for the MICRO^{3}C.

Operand numbers for calendar and clock are allocated as listed on the right:

Calendar/clock operand number	Data
0000	Year
0001	Month
0002	Day
0003	Day of week
0004	Hour
0005	Minute
0006	Second

When the range specified by the data type and data length is invalid, MICRO^3 returns an NG reply.

When a data register is designated as a preset value for a timer or counter, data cannot be written into the preset value. To change the preset value, write data into the data register designated as a preset value.

Reply Messages (Write N Bytes)

OK Reply

06h	** **	30h	** **	0Dh
(1)	(2)	(3)	(4)	(5)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**BCC**	2 bytes	00 - 7F	Block check character
(5)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

NG Reply

06h	** **	32h	30h 3*h	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	2 (32h)	NG
(4)	**NG code**	2 bytes	06 (30h 36h) 07 (30h 37h) 08 (30h 38h)	Data range error Timer/counter preset value change error Calendar/clock data error
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Data Format in the Request Message (Write N Bytes)

X (Input), Y (Output), M (Internal Relay), and R (Shift Register)

To write ON/OFF statuses of bit operands such as inputs, outputs, internal relays, or shift registers, divide the operand numbers into 8-bit (1-byte) groups and convert the 8-bit value into a hexadecimal number.

Example: To write data to outputs Q0 through Q17 to set Q5, Q7, Q12, and Q15 and reset other outputs.

Q7							Q0
1	0	1	0	0	0	0	0
A0h							

Q17							Q10
0	0	1	0	0	1	0	0
24h							

The character array resulted from conversion into hexadecimal numbers must be sent. In this example, include data A024 (41h 30h 32h 34h) in the request message.

The data length of this example is 16 bits, or 2 (02h) bytes. So, include data length code 30h 32h in the request message.

T (Timer Preset Value), C (Counter Preset Value), and D (Data Register)

To write word operands such as timers, counters, and data registers, convert the hexadecimal values into character arrays.

Example: To send 123Bh and 4567h to data registers D0 and D1, respectively.

D0	D1
123Bh	4567h

In this example, send data 123B4567 (31h 32h 33h 42h 34h 35h 36h 37h).

The data length of this example is 2 words, or 4 (04h) bytes. So, include data length code 30h 34h in the request message.

Example: To write decimal 987 and 6543 to preset values for timers T0 and T1, respectively.

T0	T1
0987	6543
03DBh	198Fh

In this example, convert the decimal values into hexadecimal values and send data 03DB198F (30h 33h 44h 42h 31h 39h 38h 46h).

The data length of this example is 2 words, or 4 (04h) bytes. So, include data length code 30h 34h in the request message.

Since MICRO^3 uses the same memory area for timers and counters, timer and counter preset values are written into the specified operand number in the same memory area. If you want to know from the computer whether the destination operand is used for timer or counter, use the procedure for reading timer/counter preset values shown on the following pages.

W (Calendar / Clock)

To send calendar/clock operands such as year, month, day, day of week, hour, minute, and second, write each one-word (2 bytes) data directly.

Day of week data format (0 through 6) is assigned as follows:

0	**1**	**2**	**3**	**4**	**5**	**6**
Sunday	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**

Example: To send calendar/clock data Friday, July 1, 1994, 13 hour, 24 minutes, 56 seconds.

Year	Month	Day	Day of week	Hour	Minute	Second
94	July	1	Friday	13	24	56
0094	0007	0001	0005	0013	0024	0056

In this example, send data 0094000700010005001300240056 (30h 30h 39h 34h 30h 30h 30h 37h 30h 30h 30h 31h 30h 30h 30h 35h 30h 30h 31h 33h 30h 30h 32h 34h 30h 30h 35h 36h).

The data length of this example is 7 words, or 14 (0Eh) bytes. So, include data length code 30h 3Eh in the request message.

Calendar/clock data cannot be written into 10-point I/O type MICRO^3 base units which do not have calendar/clock functions.

Read N Bytes

Data can be read from N-bytes of operands starting with the specified operand number in the MICRO^3 base unit.

Like the monitor mode using the program loader, this command can be used to monitor the ON/OFF statuses of bit operands such as inputs, outputs, internal relays, and shift register bits in units of 8 bits.

This command can also be used to monitor preset and current values of timers and counters, data of data registers, and read data of calendar and clock (FUN28).

Request Message (Read N Bytes)

05h	** **	30h	52h	**	** ** ** **	** **	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**Command**	1 byte	R (52h)	Read data
(5)	**Data type**	1 byte	See table below.	N-byte designation
(6)	**Operand number**	4 bytes	See table below.	First operand number to read
(7)	**Data length**	2 bytes	00 - C8	Byte count of data to read 200 (C8h) bytes maximum
(8)	**BCC**	2 bytes	00 - 7F	Block check character
(9)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

(5) Data type code		(6) Operand number		Remarks
		High-speed processing	**Standard processing**	
X (58h)	Input	0000 - 0017	0000 - 0037	The least significant digit of the operand number is an octal number (0 through7). Upper digits are decimal numbers.
Y (59h)	Output	0000 - 0017	0000 - 0037	
M (4Dh)	Internal relay	0000 - 0047 0290 - 0317	0000 - 0287 0290 - 0317	
R (52h)	Shift register	0000 - 0031	0000 - 0063	All four digits of the operand number are decimal numbers.
T (54h)	Timer (preset value)	0000 - 0015	0000 - 0031	
t (74h)	Timer (current value)	0000 - 0015	0000 - 0031	
C (43h)	Counter (preset value)	0000 - 0015	0000 - 0031	
c (63h)	Counter (current value)	0000 - 0015	0000 - 0031	
D (44h)	Data register	0000 - 0031	0000 - 0099*	
W (57h)	Calendar/clock	0000 - 0006	0000 - 0006	

Note*: Data registers can be up to 0099 for the MICRO^3 and up to 0499 for the MICRO^{3}C.

Operand numbers for calendar and clock are allocated as listed on the right:

Calendar/clock operand number	Data
0000	Year
0001	Month
0002	Day
0003	Day of week
0004	Hour
0005	Minute
0006	Second

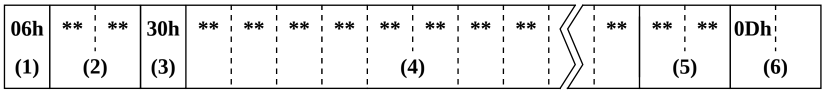
The internal relay memory area is divided into the ordinary internal relays and special internal relays. N-byte data cannot be read from the internal relay area continuing from the ordinary internal relays through special internal relays.

When the range specified by the data type and data length is invalid, MICRO^3 returns an NG reply.

When a preset value is read from a timer or counter for which a data register is designated as a preset value, the data register number is returned as a reply.

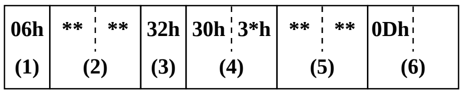
Reply Messages (Read N Bytes)

OK Reply



(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**Data**	Variable length	0 (30h) - 9 (39h) A (41h) - F (46h)	Read data
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

NG Reply



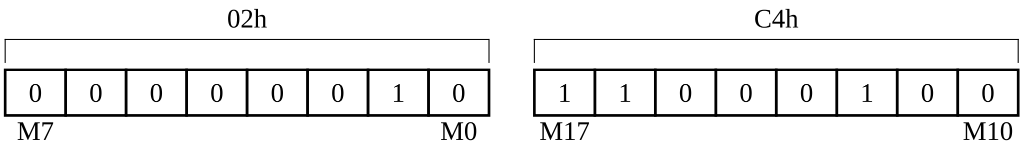
(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	2 (32h)	NG
(4)	**NG code**	2 bytes	06 (30h 36h) 08 (30h 38h)	Data range error Calendar/clock data error
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Data Format in the Reply Message (Read N Bytes)

X (Input), Y (Output), M (Internal Relay), and R (Shift Register)

When reading ON/OFF statuses of bit operands such as inputs, outputs, internal relays, or shift registers, the received data show the hexadecimal value of 8-bit groups.

Example: The read data is 02C4 when reading 2 bytes starting with internal relay M0.

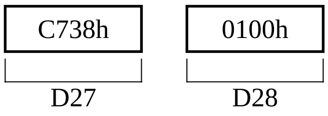


Divide the read data into one-byte (8-bit) groups. The bits where a 1 is stored are ON. In this example, internal relays M1, M12, M16, and M17 are on.

D (Data Register)

When reading data registers, the received data show the hexadecimal values in four characters each.

Example: The read data is C7380100 when reading 4 bytes starting with data register D27.



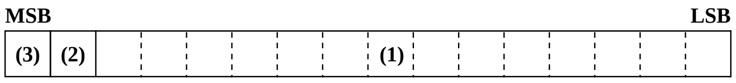
Divide the received data into 4-character groups and convert the data into 4-digit hexadecimal values. In this example, the read data is shown below:

D27 = C738h (51000 decimal)
D28 = 100h (256 decimal)

T (Timer Preset Value) and C (Counter Preset Value)

Timer/counter preset values are received in units of 2 bytes with the internal data structure as shown below:

Timer/Counter Preset Value Data Format

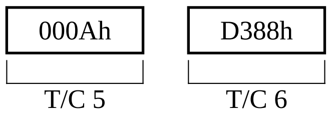


(1)	**Preset value**	0 through 9999 (0000h through 270Fh)			
(2)	**Preset value operand type**	0	Data register	1	Constant
(3)	**Timer or Counter**	0	Timer	1	Counter

Since MICRO3 uses the same memory area for timers and counters, timer and counter preset values are read from the specified operand number in the same memory area.

Example: The read data is 000AD388 when reading 4 bytes starting with timer T5 for reading timer preset values.

Divide the received data into 4-character groups and convert the data into 4-digit hexadecimal values.



To determine whether the read data is for a timer or counter, AND the 4-digit hexadecimal value with 8000h to see if the MSB is 0 or 1.

When the result is 0000h, the MSB is 0, which means the preset value is for a timer.

When the result is 8000h, the MSB is 1, which means the preset value is for a counter.

T/C 5 $\boxed{000Ah}$ AND $\boxed{8000h} \longrightarrow \boxed{0000h}$ Timer

T/C 6 $\boxed{D388h}$ AND $\boxed{8000h} \longrightarrow \boxed{8000h}$ Counter

In this example, number 5 is a timer (T5) and number 6 is a counter (C6).

Next, to determine whether the preset value is designated with a data register or constant, AND the 4-digit hexadecimal value with 4000h to see if the second MSB (bit 14) is 0 or 1.

When the result is 0000h, the bit is 0, which means the preset value is a data register.

When the result is 4000h, the bit is 1, which means the preset value is a constant.

T5 $\boxed{000Ah}$ AND $\boxed{4000h} \longrightarrow \boxed{0000h}$ Data register

C6 $\boxed{D388h}$ AND $\boxed{4000h} \longrightarrow \boxed{4000h}$ Constant

In this example, the T5 preset value is a data register and the C6 preset value is a constant.

To determine the preset value, AND the 4-digit hexadecimal value with 3FFFh to mask off upper 2 bits.

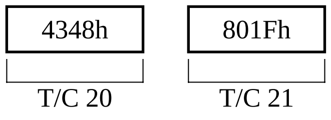
In this example, the timer T5 preset value is the data in data register D000Ah (D10) and the C6 preset value is 1388h (5000).

T5 $\boxed{000Ah}$ AND $\boxed{3FFFh} \longrightarrow \boxed{000Ah}$ D10

C6 $\boxed{D388h}$ AND $\boxed{3FFFh} \longrightarrow \boxed{1388h}$ 5000

Example: The read data is 4348801F when reading 4 bytes starting with counter C20 for reading counter preset values.

Divide the received data into 4-character groups and convert the data into 4-digit hexadecimal values.



To determine whether the read data is for a timer or counter, AND the 4-digit hexadecimal value with 8000h to see if the MSB is 0 or 1.

When the result is 0000h, the MSB is 0, which means the preset value is for a timer.

When the result is 8000h, the MSB is 1, which means the preset value is for a counter.

T/C 20 $\boxed{4348h}$ AND $\boxed{8000h} \longrightarrow \boxed{0000h}$ Timer

T/C 21 $\boxed{801Fh}$ AND $\boxed{8000h} \longrightarrow \boxed{8000h}$ Counter

In this example, number 20 is a timer (T20) and number 21 is a counter (C21).

Next, to determine whether the preset value is designated with a data register or constant, AND the 4-digit hexadecimal value with 4000h to see if the second MSB (bit 14) is 0 or 1.

When the result is 0000h, the bit is 0, which means the preset value is a data register.

When the result is 4000h, the bit is 1, which means the preset value is a constant.

T20	4348h	AND	4000h	→	4000h	Constant
C21	801Fh	AND	4000h	→	0000h	Data register

In this example, the T20 preset value is a constant and the C21 preset value is a data register.

To determine the preset value, AND the 4-digit hexadecimal value with 3FFFh to mask off upper 2 bits.

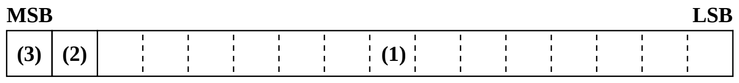
In this example, the timer T20 preset value is 0348h (840) and the C21 preset value is the data in data register D001Fh (D31).

T20	4348h	AND	3FFFh	→	0348h	840
C21	801Fh	AND	3FFFh	→	001Fh	D31

t (Timer Current Value) and c (Counter Current Value)

Timer/counter current values are received in units of 2 bytes with the internal data structure as shown below:

Timer/Counter Current Value Data Format



(1)	**Current value**	0 through 9999 (0000h through 270Fh)			
(2)	**(Reserved)**	Indefinite			
(3)	**Timeout or countout flag**	0	Not timeout or countout (OFF)	1	Timeout or countout (ON)

Since MICRO3 uses the same memory area for timers and counters, timer and counter current values are read from the specified operand number in the same memory area. If you want to know the source operand is used for timer or counter from the computer, use the procedure for reading timer/counter preset values shown on the preceding page.

Example: The read data is 03E88037 when reading 4 bytes starting with timer T13 for reading timer current values.

Divide the received data into 4-character groups and convert the data into 4-digit hexadecimal values.

03E8h	8037h
T13	T14

To determine the current value, AND the 4-digit hexadecimal value with 3FFFh to mask off upper 2 bits.

In this example, the timer T13 current value is 03E8h (1000) and the timer T14 current value is 0037h (55).

T13	03E8h	AND	3FFFh	→	03E8h	1000
T14	8037h	AND	3FFFh	→	0037h	55

To determine whether the timer has been timed out or not, AND the 4-digit hexadecimal value with 8000h to see if the MSB is 0 or 1.

When the result is 0000h, the MSB is 0, which means the timer is not timed out yet and the status is OFF.

When the result is 8000h, the MSB is 1, which means the timer has been timed out and the status is ON.

T13	03E8h	AND	8000h	→	0000h	Not timed out (OFF)
T14	8037h	AND	8000h	→	8000h	Timed out (ON)

In this example, timer T13 is not timed out yet (OFF) and timer T14 has been timed out (ON).

Example: The read data is 119E974B when reading 4 bytes starting with counter C25 for reading counter current values.

The counter current values are also processed in the same manner.

Divide the received data into 4-character groups and convert the data into 4-digit hexadecimal values.

119Eh	974Bh
C25	C26

To determine the current value, AND the 4-digit hexadecimal value with 3FFFh to mask off upper 2 bits.

In this example, the counter C25 current value is 119Eh (4510) and the counter C26 current value is 174Bh (5963).

C25 $\boxed{\text{119Eh}}$ AND $\boxed{\text{3FFFh}}$ $\longrightarrow$ $\boxed{\text{119Eh}}$ 4510

C26 $\boxed{\text{974Bh}}$ AND $\boxed{\text{3FFFh}}$ $\longrightarrow$ $\boxed{\text{174Bh}}$ 5963

To determine whether the counter has been counted out or not, AND the 4-digit hexadecimal value with 8000h to see if the MSB is 0 or 1.

When the result is 0000h, the MSB is 0, which means the counter is not counted out yet and the status is OFF.

When the result is 8000h, the MSB is 1, which means the counter has been counted out and the status is ON.

C25 $\boxed{\text{119Eh}}$ AND $\boxed{\text{8000h}}$ $\longrightarrow$ $\boxed{\text{0000h}}$ Not counted out (OFF)

C26 $\boxed{\text{974Bh}}$ AND $\boxed{\text{8000h}}$ $\longrightarrow$ $\boxed{\text{8000h}}$ Counted out (ON)

In this example, counter C25 is not counted out yet (OFF) and counter C26 has been counted out (ON).

W (Calendar / Clock)

Calendar/clock data are received in units of 2 bytes starting with the specified operand number 0000 (year) through 0006 (second). For operand numbers for the calendar and clock, see page 4-8.

Day of week data format (0 through 6) is assigned as follows:

0	**1**	**2**	**3**	**4**	**5**	**6**
Sunday	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**

Calendar/clock data cannot be read from 10-point I/O type MICRO^3 base units which do not have calendar/clock functions.

Example: The read data is 000200200059 when reading 6 bytes (3 words) starting with operand number 0003 (day of week) for reading calendar/clock values.

Divide the received data into 4-character groups and convert the data into 4-digit hexadecimal values.

Data of three operands starting with 0003 (day of week) is read as shown on the right.

$\underbrace{\boxed{\text{0002h}}}_{\text{2 = Tuesday}}$ $\underbrace{\boxed{\text{0020h}}}_{\text{20 hours}}$ $\underbrace{\boxed{\text{0059h}}}_{\text{59 minutes}}$

Write 1 Bit

Data can be written into 1 bit of the specified operand in the MICRO^3 base unit, enabling to set (ON) or reset (OFF) the operand.

MICRO^3 operation can be started or stopped by setting or resetting start control special internal relay M300 using this request message.

The write 1 bit command has the same function as the setting and resetting operation using the program loader.

Request Message (Write 1 Bit)

05h	**	**	30h	57h	**	**	**	**	**	3*h	**	**	0Dh	
(1)	(2)		(3)	(4)	(5)		(6)			(7)	(8)		(9)	

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**Command**	1 byte	W (57h)	Write data
(5)	**Data type**	1 byte	See table below.	1-bit designation
(6)	**Operand number**	4 bytes	See table below.	Operand number to write to
(7)	**ON / OFF status**	1 byte	0 (30h) 1 (31h)	OFF ON
(8)	**BCC**	2 bytes	00 - 7F	Block check character
(9)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

(5) Data type code		(6) Operand number		Remarks
		High-speed processing	Standard processing	
x (78h)	Input	0000 - 0017	0000 - 0037	The least significant digit of the operand number is an octal number (0 through7). Upper digits are decimal numbers.
y (79h)	Output	0000 - 0017	0000 - 0037	
m (6Dh)	Internal relay	0000 - 0047 0290 - 0317	0000 - 0287 0290 - 0317	
r (72h)	Shift register	0000 - 0031	0000 - 0063	

Reply Messages (Write 1 Bit)

OK Reply

06h	**	**	30h	**	**	0Dh	
(1)	(2)		(3)	(4)		(5)	

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**BCC**	2 bytes	00 - 7F	Block check character
(5)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

NG Reply

06h	**	**	32h	30h	36h	**	**	0Dh	
(1)	(2)		(3)	(4)		(5)		(6)	

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	2 (32h)	NG
(4)	**NG code**	2 bytes	06 (30h 36h)	Data range error
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Read 1 Bit

Data can be read from 1 bit of the specified operand in the MICRO^3 base unit to see if the operand is on or off.

The read 1 bit command can be used to monitor the ON/OFF status of a bit operand such as input, output, internal relay, or shift register bit.

Request Message (Read 1 Bit)

05h	**	**	30h	52h	**	**	**	**	**	**	**	0Dh	
(1)	(2)		(3)	(4)	(5)	(6)				(7)		(8)	

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**Command**	1 byte	R (52h)	Read data
(5)	**Data type**	1 byte	See table below.	1-bit designation
(6)	**Operand number**	4 bytes	See table below.	Operand number to read from
(7)	**BCC**	2 bytes	00 - 7F	Block check character
(8)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

(5) Data type code		(6) Operand number		Remarks
		High-speed processing	**Standard processing**	
x (78h)	Input	0000 - 0017	0000 - 0037	The least significant digit of the operand number is an octal number (0 through7). Upper digits are decimal numbers.
y (79h)	Output	0000 - 0017	0000 - 0037	
m (6Dh)	Internal relay	0000 - 0047 0290 - 0317	0000 - 0287 0290 - 0317	
r (72h)	Shift register	0000 - 0031	0000 - 0063	

Reply Messages (Read 1 Bit)

OK Reply

06h	**	**	30h	3*h	**	**	0Dh
(1)	(2)		(3)	(4)	(5)		(6)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**ON / OFF status**	1 byte	0 (30h) 1 (31h)	OFF ON
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

NG Reply

06h	**	**	32h	30h	36h	**	**	0Dh
(1)	(2)		(3)	(4)		(5)		(6)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	2 (32h)	NG
(4)	**NG code**	2 bytes	06 (30h 36h)	Data range error
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Read High-speed Counter Preset and Current Values

Preset and current values of high-speed counters HSC0 through HSC3 can be read from the MICRO^3 base unit.

This command can be used to monitor the preset and current values of high-speed counters.

Request Message (Read High-speed Counter Preset and Current Values)

05h	**	**	30h	52h	55h	**	**	**	**	**	**	0Dh
(1)	(2)		(3)	(4)	(5)	(6)				(7)		(8)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**Command**	1 byte	R (52h)	Read data
(5)	**Data type**	1 byte	U (55h)	High-speed counter (preset and current values)
(6)	**High-speed counter number**	4 bytes	0000 0001 0002 0003	HSC0 HSC1 HSC2 HSC3
(7)	**BCC**	2 bytes	00 - 7F	Block check character
(8)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Reply Messages (Read High-speed Counter Preset and Current Values)

OK Reply

06h	**	**	30h	3*h	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	**	0Dh
(1)	(2)		(3)	(4)	(5)								(6)								(7)		(8)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**Preset value operand type**	1 byte	0 (30h) 1 (31h) 2 (32h)	Data register (variable) Constant Specified HSC number not found
(5)	**Preset value data**	8 bytes	00000000 - FFFFFFFF	HSC preset value
(6)	**Current value data**	8 bytes	00000000 - FFFFFFFF	HSC current value
(7)	**BCC**	2 bytes	00 - 7F	Block check character
(8)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

When an OK reply is returned with (4) preset value operand type containing “0” (data register), the upper 4 bytes of (5) preset value data contain always 0000 in ▭. The lower 4 bytes indicate the first data register number designated for the high-speed counter preset value, which uses two consecutive data registers containing upper and lower digits, respectively.

When the high-speed counter number designated by (6) high-speed counter number in the request message is not found in MICRO^3, the reply message indicates “2” (specified HSC number not found) in place of (4) preset value operand type and contains 00000000 in both preset and current values.

NG Reply

06h	**	**	32h	30h	36h	**	**	0Dh
(1)	(2)		(3)	(4)		(5)		(6)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	2 (32h)	NG
(4)	**NG code**	2 bytes	06 (30h 36h)	Data range error
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Read Error Code

Error codes can be read from the MICRO^3 base unit.

This function is the same as FUN20 to read error data using the program loader.

Request Message (Read Error Code)

05h	**	**	30h	52h	45h	30h	30h	30h	3*h	30h	**	**	**	0Dh
(1)	(2)		(3)	(4)	(5)	(6)				(7)		(8)		(9)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**Command**	1 byte	R (52h)	Read data
(5)	**Data type**	1 byte	E (45h)	Error code
(6)	**Error address**	4 bytes	See table below.	First error address to read
(7)	**Data length**	2 bytes	00 - 0C	2 bytes per error address
(8)	**BCC**	2 bytes	00 - 7F	Block check character
(9)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

(6) Error address	Error details
0000	Error code
0001	User program syntax error: Type code
0002	User program syntax error: Address code
0003	Advanced instruction syntax error
0004	User program execution error
0005	Link communication error

Reply Messages (Read Error Code)

OK Reply

06h	**	**	30h	**	**	**	**	**	**	**	**	**	** **	0Dh
(1)	(2)		(3)				(4)						(5)	(6)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**Data**	Variable length	0 (30h) - 9 (39h) A (41h) - F (46h)	Error code
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

NG Reply

06h	** **	32h	30h 36h	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	2 (32h)	NG
(4)	**NG code**	2 bytes	06 (30h 36h)	Data range error (error address)
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Data Format in the Reply Message (Read Error Code)

When reading error codes, the received data show the hexadecimal values in four characters each.

Example: The read data is 0080 0006 0004 0001 0000 0000 when reading 12 (0Ch) bytes starting with error address 0000.

Divide the received data into 4-character groups and convert the data into 4-digit hexadecimal values.

Error address:	0000	0001	0002	0003	0004	0005
	0080h	0006h	0004h	0001h	0000h	0000h

In this example, the read data is shown below:

0000	Error code	80h
0001	User program syntax error: Type code	6h
0002	User program syntax error: Address code	4h
0003	Advanced instruction syntax error	1h
0004	User program execution error	0h
0005	Link communication error	0h

The above data means that user program syntax error (error code 80h) is found.
The type code of the user program syntax error is 6h (invalid data for advanced instruction).
The address code of the user program syntax error is 4h (address 4).
The advanced instruction syntax error code is 1h which means that the internal allocation number of the operand is invalid.

For details of error codes, see MICRO³ User's Manual EM317.

Clear Operand Data

All data of selected operand area or all operands can be cleared from the MICRO^3 base unit.

This command is the same as FUN26 to clear operand and FUN27 to execute the link formatting sequence using the program loader.

Request Message (Clear Operand Data)

05h	** **	30h	43h	**	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)	(7)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**Command**	1 byte	C (43h)	Clear data
(5)	**Data type**	1 byte	See table below.	
(6)	**BCC**	2 bytes	00 - 7F	Block check character
(7)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

(5) Data type	Data to clear	(5) Data type	Data to clear
X (58h)	Input	**C (43h)**	Counter (preset value)
Y (59h)	Output	**c (63h)**	Counter (current value)
M (4Dh)	Internal relay	**D (44h)**	Data register
R (52h)	Shift register	**E (45h)**	Error code
T (54h)	Timer (preset value)	**Z (5Ah)**	System reset (all operands)
t (74h)	Timer (current value)	**I (49h)**	Link formatting sequence

When the timer preset value (T) or counter preset value (C) is cleared, the changed preset values in the MICRO^3 base unit RAM are cleared and the original preset values are restored.

When the system reset is executed with Z (5Ah) specified for the (5) data type, data is cleared from all operand areas of inputs (X), outputs (Y), internal relays (M), shift registers (R), timer current values (t), counter current values (c), and data registers (D).

When the link formatting sequence (I) is executed, the data link terminal connection data is updated like executing FUN27 using the program loader.

Reply Messages (Clear Operand Data)

OK Reply

06h	** **	30h	** **	0Dh
(1)	(2)	(3)	(4)	(5)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**BCC**	2 bytes	00 - 7F	Block check character
(5)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

NG Reply

06h	** **	32h	30h 39h	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	2 (32h)	NG
(4)	**NG code**	2 bytes	09 (30h 39h)	Data clear error
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Read PLC Operating Status

This command can be used to read the operating status of the MICRO^3 base unit to the computer. When this command is executed, the received data also indicates whether the timer/counter preset values have been changed, whether the user program in MICRO^3 is protected, and the type of the MICRO^3 base unit.

Similar reading functions are allocated to FUN21 (timer/counter preset value readout), FUN22 (user program protection), and FUN24 (PLC operating status readout) executed using the program loader.

Request Message (Read PLC Operating Status)

05h	**	**	30h	52h	53h	**	**	0Dh	
(1)	(2)		(3)	(4)	(5)	(6)		(7)	

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**Command**	1 byte	R (52h)	Read data
(5)	**Data type**	1 byte	S (53h)	PLC operating status
(6)	**BCC**	2 bytes	00 - 7F	Block check character
(7)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Reply Message (Read PLC Operating Status)

OK Reply

06h	** ¦ **	30h	3*h	3*h	3*h	3*h ¦ **	** ¦ **	0Dh ¦
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**PLC operating status**	1 byte	0 (30h) 1 (31h)	Run Stop
(5)	**Timer / counter preset value change**	1 byte	0 (30h) 1 (31h)	Not changed Changed
(6)	**User program protection**	1 byte	0 (30h) 1 (31h) 2 (32h) 3 (33h)	Not protected Write protect Read protect Read and write protect
(7)	**MICRO^3 base unit type code**	2 bytes	See table below.	MICRO^3 base unit type
(8)	**BCC**	2 bytes	00 - 7F	Block check character
(9)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

(7) MICRO^3 base unit type code	Input type		Output protection		I / O points		
	DC	AC	Not	Protected	14 / 10	9 / 7	6 / 4
01 (30h 31h)	x		x		x		
02 (30h 32h)	x		x			x	
04 (30h 34h)	x		x				x
11 (31h 31h)	x			x	x		
12 (31h 32h)	x			x		x	
14 (31h 34h)	x			x			x
0A (30h 41h)		x	x			x	

Note: NG replies are not returned in response to the request message of reading the PLC operating status.

Read Scan Time

The scan time of the user program in operation can be read from the MICRO3 base unit. When this command is executed, the received data indicates the current and maximum values of the user program scan time.

This command has the same function as FUN25 scan time readout using the program loader.

Request Message (Read Scan Time)

05h	** **	30h	52h	4Bh	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)	(7)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**Command**	1 byte	R (52h)	Read data
(5)	**Data type**	1 byte	K (4Bh)	Scan time
(6)	**BCC**	2 bytes	00 - 7F	Block check character
(7)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Reply Message (Read Scan Time)

OK Reply

06h	**	**	30h	**	**	**	**	**	**	**	**	**	**	0Dh
(1)	(2)		(3)	(4)				(5)				(6)		(7)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**Scan time (current value)**	4 bytes	0000 - FFFF	Current value of the scan time
(5)	**Scan time (maximum value)**	4 bytes	0000 - FFFF	Maximum value of the scan time
(6)	**BCC**	2 bytes	00 - 7F	Block check character
(7)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Note: NG replies are not returned in response to the request message of reading the scan time.

Data Format in the Reply Message (Read Scan Time)

The scan time is read in units of msec.

The current and maximum values of the scan time are displayed in the hexadecimal notation for the integer and in the octal notation for the fraction.

Example: The read data is 0145 when reading the scan time.

Divide the received data into upper 3 characters and the lowest 1 character.

014h	5o

The upper 3 digits indicate the integer part of the scan time in the hexadecimal notation.
The lowest digit indicates the fraction part of the scan time in the octal notation.

In this example, the scan time reads $1 \times 16 + 4 + 0.5 \times 1.25 = 20.625$ msec in the decimal notation.

Read PLC System Program Version

The system program version of the MICRO^3 base unit can be read to the computer.

This command has the same function as FUN23 PLC system program version readout using the program loader.

Request Message (Read PLC System Program Version)

05h	** **	30h	52h	4Eh	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)	(7)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**Command**	1 byte	R (52h)	Read data
(5)	**Data type**	1 byte	N (4Eh)	PLC system program version
(6)	**BCC**	2 bytes	00 - 7F	Block check character
(7)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Reply Message (Read PLC System Program Version)

OK Reply

06h	** **	30h	** ** ** **	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**PLC system program version**	4 bytes	0000 - FFFF	System program version of MICRO^3
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Note: NG replies are not returned in response to the request message of reading the PLC system program version.

Data Format in the Reply Message (Read PLC System Program Version)

The PLC system program version is read in the hexadecimal notation.

Example: The read data is 0123 when reading the PLC system program version.

The PLC program version is 0123.

Read User Communication Transmit / Receive Buffer

◆ MICRO^{3}C Only ◆

While user communication is performed through the loader port, the transmit and receive buffers store the data of the last communication. The data stored in the transmit and receive buffers can be read through the data link terminals using the read user communication transmit/receive buffer command.

This command can be used on the MICRO^{3}C only.

Request Message (Read User Communication Transmit / Receive Buffer)

05h	** **	30h	52h	**	** ** ** **	** **	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**Command**	1 byte	R (52h)	Read data
(5)	**Data type**	1 byte	G (47h) g (67h)	User communication receive buffer User communication transmit buffer
(6)	**Operand number**	4 bytes	0000 - 0199	First address to read data (Nth byte in the 200-byte buffer)
(7)	**Data length**	2 bytes	01 - C9	Byte count of data to read 200 (C8h) bytes maximum + 1h
(8)	**BCC**	2 bytes	00 - 7F	Block check character
(9)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

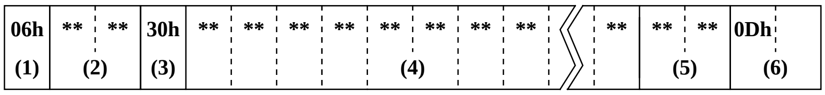
(7) Data length

Specify the byte count of the data to read plus one as a data length. Since a 2-byte ASCII code is attached to the beginning of the data codes in the reply message, one byte must be added to the data length in the request message. The additional 2-byte ASCII code represents the byte count of read data (see the next page).

The transmit/receive buffer has a capacity of 200 (C8h) bytes. When reading the entire data in the transmit/receive buffer, specify C9 (43h 39h) as a data length.

Reply Messages (Read User Communication Transmit / Receive Buffer)

OK Reply

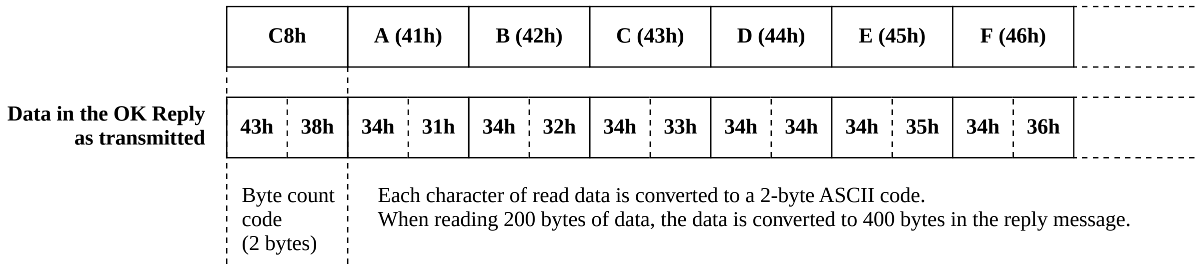


(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**Data**	Variable length 402 bytes max.	0 (30h) - 9 (39h) A (41h) - F (46h)	Read data byte count and read data
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

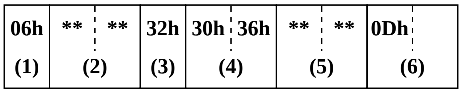
(4) Data

The byte count of data read from the transmit/receive buffer is stored in the first 2 bytes of the data codes, represented in ASCII code. The read data is stored starting at the third byte in the data codes, also represented in ASCII code.

Example: The byte count of data to read is C8h (maximum value of 200 bytes).
The data in the transmit/receive buffer is ABCEDF ...... (41h 42h 43h 44h 45h 46h ......)


C8h
A (41h)
B (42h)
C (43h)
D (44h)
E (45h)
F (46h)
Data in the OK Reply as transmitted
43h 38h
34h 31h
34h 32h
34h 33h
34h 34h
34h 35h
34h 36h
Byte count code (2 bytes)
Each character of read data is converted to a 2-byte ASCII code.
When reading 200 bytes of data, the data is converted to 400 bytes in the reply message.

NG Reply



(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	2 (32h)	NG
(4)	**NG code**	2 bytes	06 (30h 36h)	Data range error
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Clear and Start User Communication Data Monitor ◆ MICRO3C Only ◆

This command is implemented in FUN50 user communication data monitor using the program loader.

While user communication is performed through the loader port, the transmit and receive data of the user communication can be monitored from the data link terminals. The data monitor can also be performed when the loader port is shifted from the modem mode to the loader protocol.

When the MICRO3C receives this command through the data link terminals, the MICRO3C clears the monitor buffer of the previous user communication data, and starts to monitor the incoming and outgoing data through the loader port. The transition to the monitor mode occurs in approximately 100 msec after receiving this command. The monitored data is stored in the monitor buffer temporarily and sent out through the data link terminals continuously.

The internal buffer for the user communication data monitor has a capacity of 256 bytes. When the buffer is filled to the full capacity, the monitor data is not updated.

Once the user communication data monitor mode is started, the monitor mode remains in effect until the communication enable button on the MICRO3C is pressed or the MICRO3C is powered up again. When a communication error, such us parity error, framing error, or overrun error, occurs during monitoring, the MICRO3C sends out an error code E! from the data link terminals.

When 8-bit data is received while FUN8 is set to select 7 data bits for user communication, the MSB of the monitored data is ignored.

This command can be used on the MICRO3C only.

Request Message (Clear and Start User Communication Data Monitor)

05h	**	**	30h	43h	4*h	**	**	0Dh
(1)	(2)		(3)	(4)	(5)	(6)		(7)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**Command**	1 byte	C (43h)	Clear data
(5)	**Data type**	1 byte	A (41h) B (42h)	19,200 bps, D8, NP, S1 9,600 bps, D8, NP, S1 (default)
(6)	**BCC**	2 bytes	00 - 7F	Block check character
(7)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

(5) Data type

Specify A (19,200 bps) or B (9,600 bps) to match the baud rate for the loader port selected in FUN8 loader port communication parameters. For either selection, other parameters are 8 data bits, none parity, and 1 stop bit.

Reply Messages (Clear and Start User Communication Data Monitor)

OK Reply

06h	**	**	30h	**	**	0Dh	
(1)	(2)		(3)	(4)		(5)	

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**BCC**	2 bytes	00 - 7F	Block check character
(5)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

NG Reply

06h	**	**	32h	31h	3*h	**	**	0Dh	
(1)	(2)		(3)	(4)		(5)		(6)	

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	2 (32h)	NG
(4)	**NG code**	2 bytes	10 (31h 30h) 11 (31h 31h)	Data error Setting error
(5)	**BCC**	2 bytes	00 - 7F	Block check character
(6)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

(7) NG code

Error code 11 is returned and the user communication data cannot be monitored in the following cases:

- The loader port is not in the user protocol mode or not in the loader protocol mode shifted from the modem mode. Note that the user communication data monitor command is accepted through the data link terminals only.
- Data type B is designated in the request message to select 9600 bps for the loader port while the loader port is set to 19,200 bps using FUN8 loader port communication mode setting.

Note: FUN50 user communication data monitor uses data type A to select 19,200 bps.

Read User Communication Status

◆ MICRO^{3}C Only ◆

This command is implemented in FUN29 user communication status readout using the program loader.

Various information concerning user communication can be read out using this command, such as the PLC operating status, user communication mode, user communication receive error interrupt, and user communication transmit/receive instruction status. Communication parameter settings are also read out. For details, see the reply message format on the next page.

This command can be used on the MICRO^{3}C only.

Request Message (Read User Communication Status)

05h	** **	30h	52h	42h	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)	(7)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**Command**	1 byte	R (52h)	Read data
(5)	**Data type**	1 byte	B (42h)	User communication status
(6)	**BCC**	2 bytes	00 - 7F	Block check character
(7)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Reply Messages (Read User Communication Status)

OK Reply

06h	** **	30h	3*h	3*h	3*h	3*h	3*h	3*h	3*h	3*h	3*h	** **	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**PLC operating status**	1 byte	0 (30h) 1 (31h)	Running Stopped
(5)	**User communication mode**	1 byte	0 (30h) 1 (31h)	User communication mode not enabled User communication mode enabled
(6)	**User communication receive error interrupt**	1 byte	0 (30h) 1 (31h)	Interrupt not occurred Interrupt occurred
(7)	**User communication receive instruction status**	1 byte	0 (30h) 2 (32h) 3 (33h)	No valid RXD instruction in user program Receiving data Waiting for executing RXD instruction
(8)	**User communication transmit instruction status**	1 byte	0 (30h) 2 (32h) 4 (34h)	No valid TXD instruction in user program Transmitting data Waiting for executing TXD instruction
(9)	**Baud rate**	1 byte	0 (30h) 1 (31h) 2 (32h) 3 (33h) 4 (34h)	1200 bps 2400 bps 4800 bps 9600 bps 19200 bps
(10)	**Data length**	1 byte	0 (30h) 1 (31h)	7 bits 8 bits
(11)	**Parity**	1 byte	0 (30h) 1 (31h) 2 (32h)	None Odd Even
(12)	**Stop bits**	1 byte	0 (30h) 1 (31h)	1 bit 2 bits
(13)	**Receive timeout**	2 bytes	01 - FF	10 to 2550 msec
(14)	**BCC**	2 bytes	00 - 7F	Block check character
(15)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

NG Reply

NG reply never occurs in response to the request message of reading the user communication status.

Read Communication Mode

◆ *MICRO³C Only* ◆

Various information concerning the MICRO³C communication functions can be read out using this command, such as the protocol selector switch position, expansion control data register service selection, modem mode selection, PLC type code, and function selector switch position. For details, see the reply message format on the next page.

This command can be used on the MICRO³C only.

Request Message (Read Communication Mode)

05h	** **	30h	52h	48h	** **	0Dh
(1)	(2)	(3)	(4)	(5)	(6)	(7)

(1)	**Communication control character**	1 byte	ENQ (05h)	Enquiry
(2)	**Communication device number**	2 bytes	00 - 1F FF	Device number 0 through 31 Device number 255 (all devices)
(3)	**Continuation**	1 byte	0 (30h)	Discontinued
(4)	**Command**	1 byte	R (52h)	Read data
(5)	**Data type**	1 byte	H (48h)	Communication mode
(6)	**BCC**	2 bytes	00 - 7F	Block check character
(7)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

Reply Messages (Read Communication Mode)

OK Reply

06h	**	**	30h	3*h	3*h	3*h	3*h	**	3*h	**	**	0Dh
(1)	(2)		(3)	(4)	(5)	(6)	(7)		(8)	(9)		(10)

(1)	**Communication control character**	1 byte	ACK (06h)	Acknowledge
(2)	**Communication device number**	2 bytes	00 - 1F	Device number 0 through 31
(3)	**Command**	1 byte	0 (30h)	OK: Discontinued
(4)	**Protocol selector switch position**	1 byte	0 (30h) to 7 (37h)	0 to 7
(5)	**Expansion Control Data Register Service Group 1 (Expansion Control D492-D495)**	1 byte	0 (30h) 1 (31h)	Group 1 disabled Group 1 enabled
(6)	**Modem mode**	1 byte	0 (30h) 1 (31h)	Modem mode disabled Modem mode enabled
(7)	**PLC type code**	2 bytes	See table below.	MICRO^{3}C base unit type
(8)	**Function selector switch position**	1 byte	0 (30h) to 7 (37h)	0 to 7
(9)	**BCC**	2 bytes	00 - 7F	Block check character
(10)	**Terminator**	1 byte 2 bytes	CR (0Dh) CR LF (0Dh 0Ah)	Message end code

(7) PLC type code	Input type		Output protection		I/O points			User communication	
	DC	AC	Not	Protected	14/10	9/7	6/4	Without	With
01 (30h 31h)	x		x		x			x	
02 (30h 32h)	x		x			x		x	
04 (30h 34h)	x		x				x	x	
11 (31h 31h)	x			x	x			x	
12 (31h 32h)	x			x		x		x	
14 (31h 34h)	x			x			x	x	
0A (30h 41h)		x	x			x		x	
21 (32h 31h)	x		x		x				x
22 (32h 32h)	x		x			x			x

NG Reply

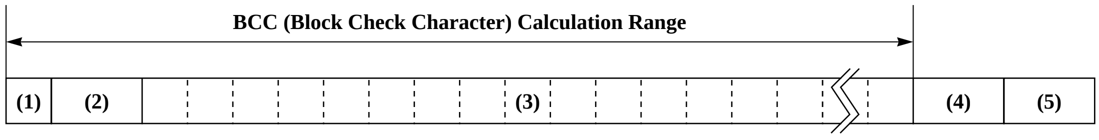
NG reply never occurs in response to the request message of reading the communication mode.

Calculating BCC

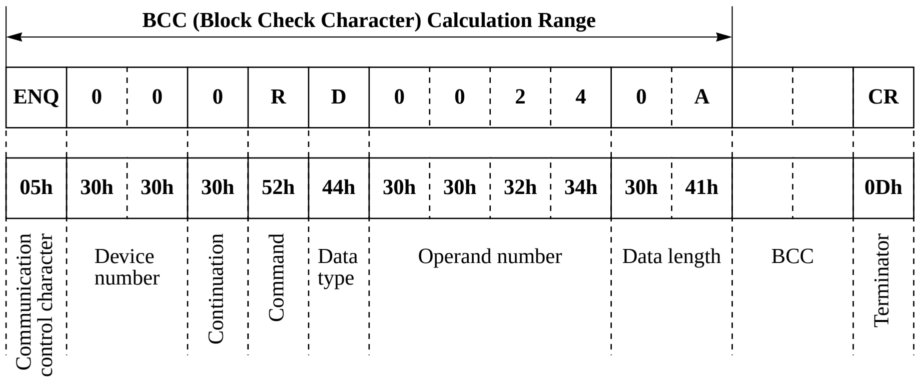
Data error may be caused by noises while the communication data is sent or received. MICRO^3 employs the BCC (block check character) horizontal parity method to detect errors in data communication.

The BCC consists of 2 bytes of characters.

The BCC is the result of exclusive OR operation (XOR) on the BCC calculation range from the first character to the character immediately before the BCC. The character string of the XOR result is inserted as a BCC before the terminator.



Example: Calculate the BCC when sending the command shown below.



The BCC is calculated by XORing hexadecimal values of characters from the communication control character up to the data length code.

05h ⊕ 30h ⊕ 30h ⊕ 30h ⊕ 52h ⊕ 44h ⊕ 30h ⊕ 30h ⊕ 32h ⊕ 34h ⊕ 30h ⊕ 41h = 54h

Therefore, the BCC is characters 5 and 4 (35h 34h).

As a result, the following request message must be sent to execute this command.

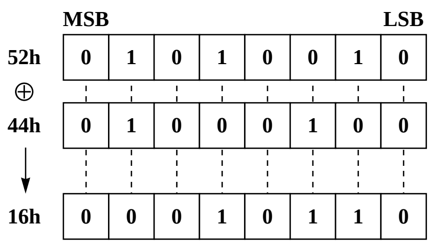
ENQ	0	0	0	R	D	0	0	2	4	0	A	5	4	CR
05h	30h	30h	30h	52h	44h	30h	30h	32h	34h	30h	41h	35h	34h	0Dh

Exclusive OR (XOR)

The exclusive OR is a logical operation which turns output on only when one of two inputs is on.

Example: 52h ⊕ 44h

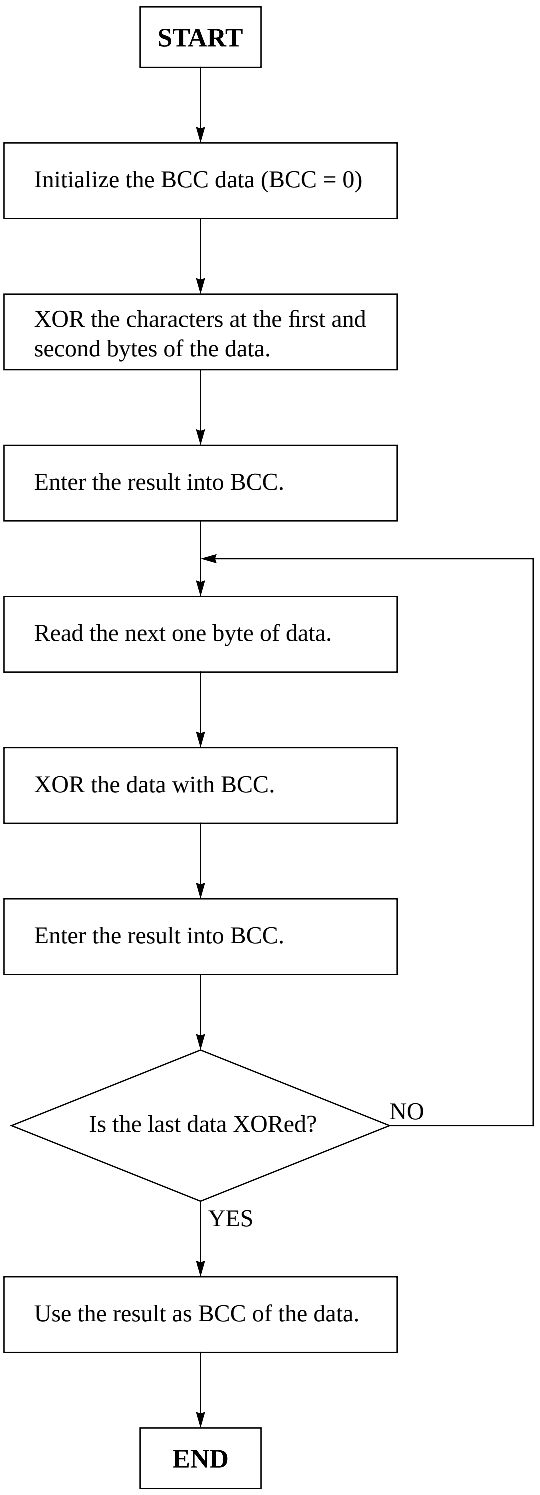
Convert both values into binary numbers and XOR each bit.



X1	X2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Flow Chart for Calculating BCC

Calculate the BCC (block check character) using the chart shown below.

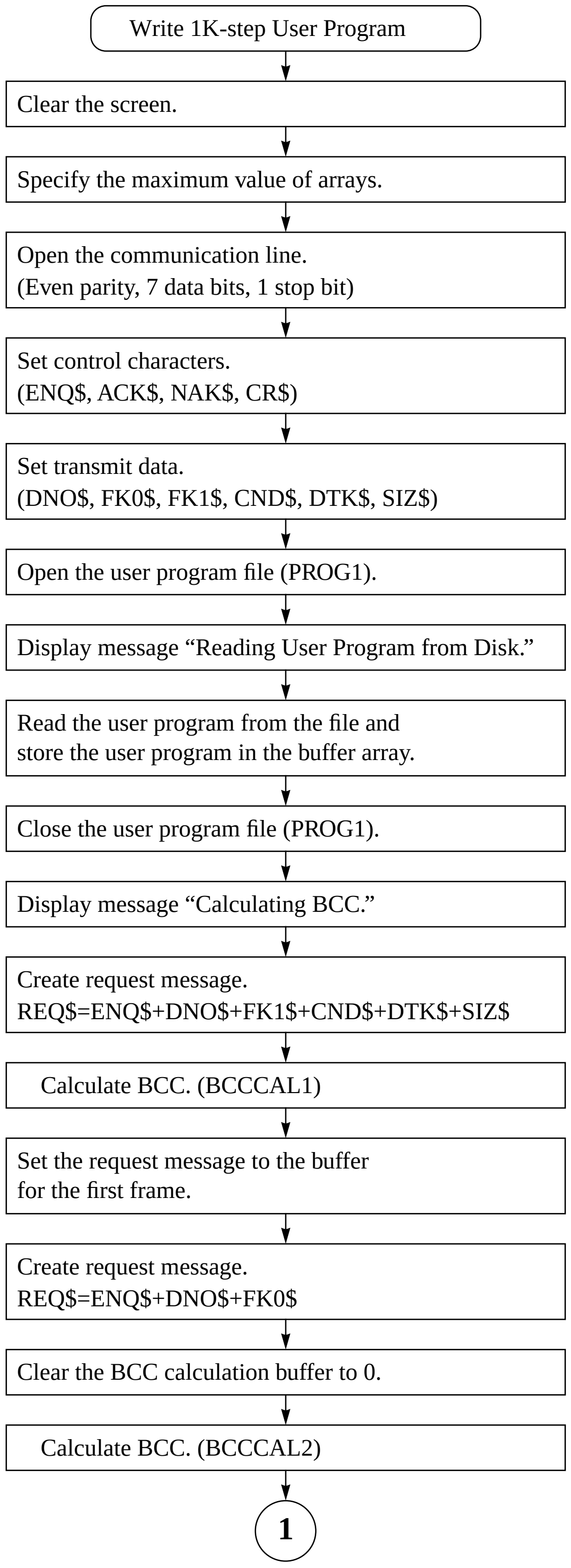

START
Initialize the BCC data (BCC = 0)
XOR the characters at the first and second bytes of the data.
Enter the result into BCC.
Read the next one byte of data.
XOR the data with BCC.
Enter the result into BCC.
Is the last data XORed?
NO
YES
Use the result as BCC of the data.
END

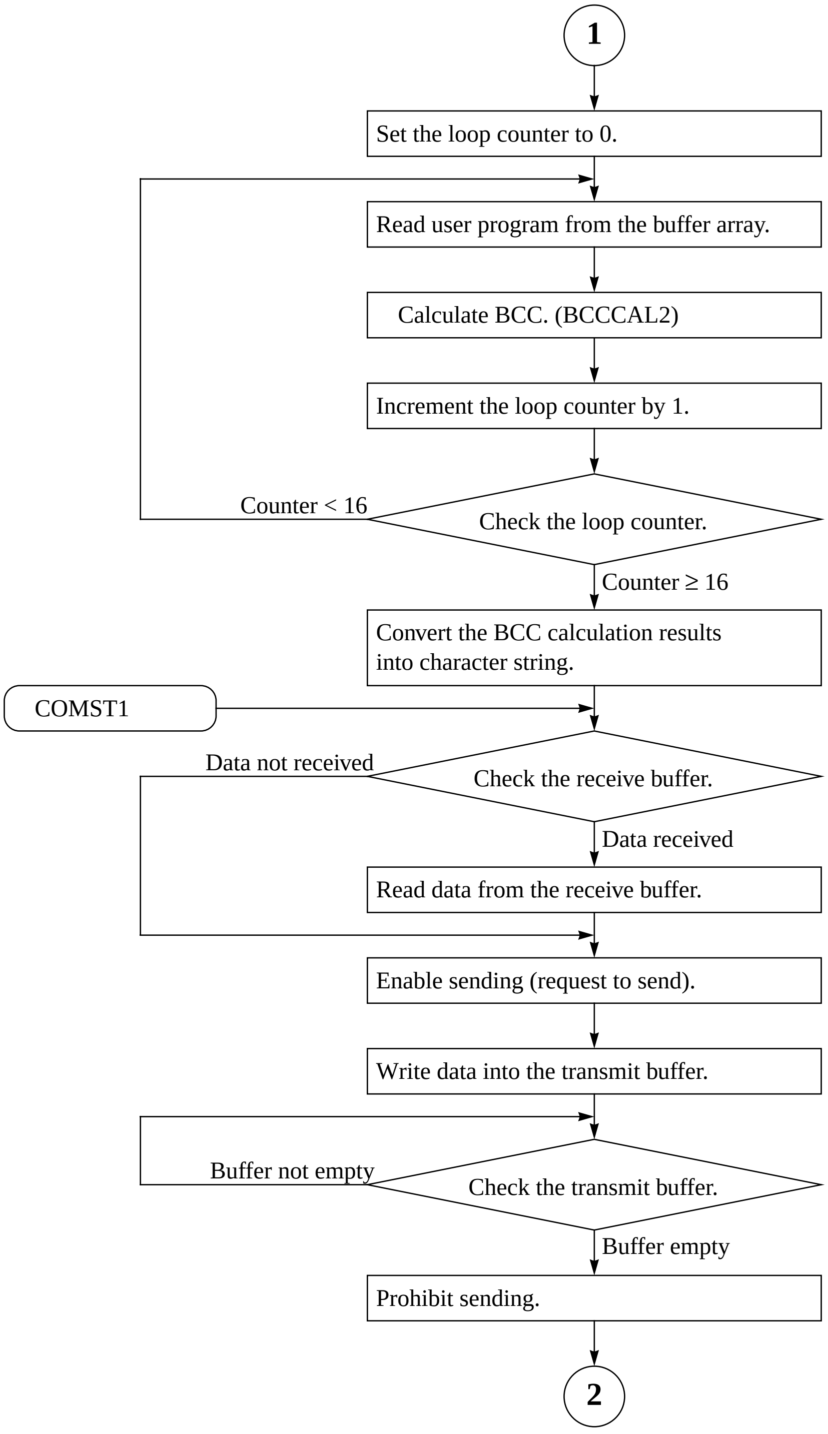
For actual programming of BCC, see sample programs shown in the following chapters.

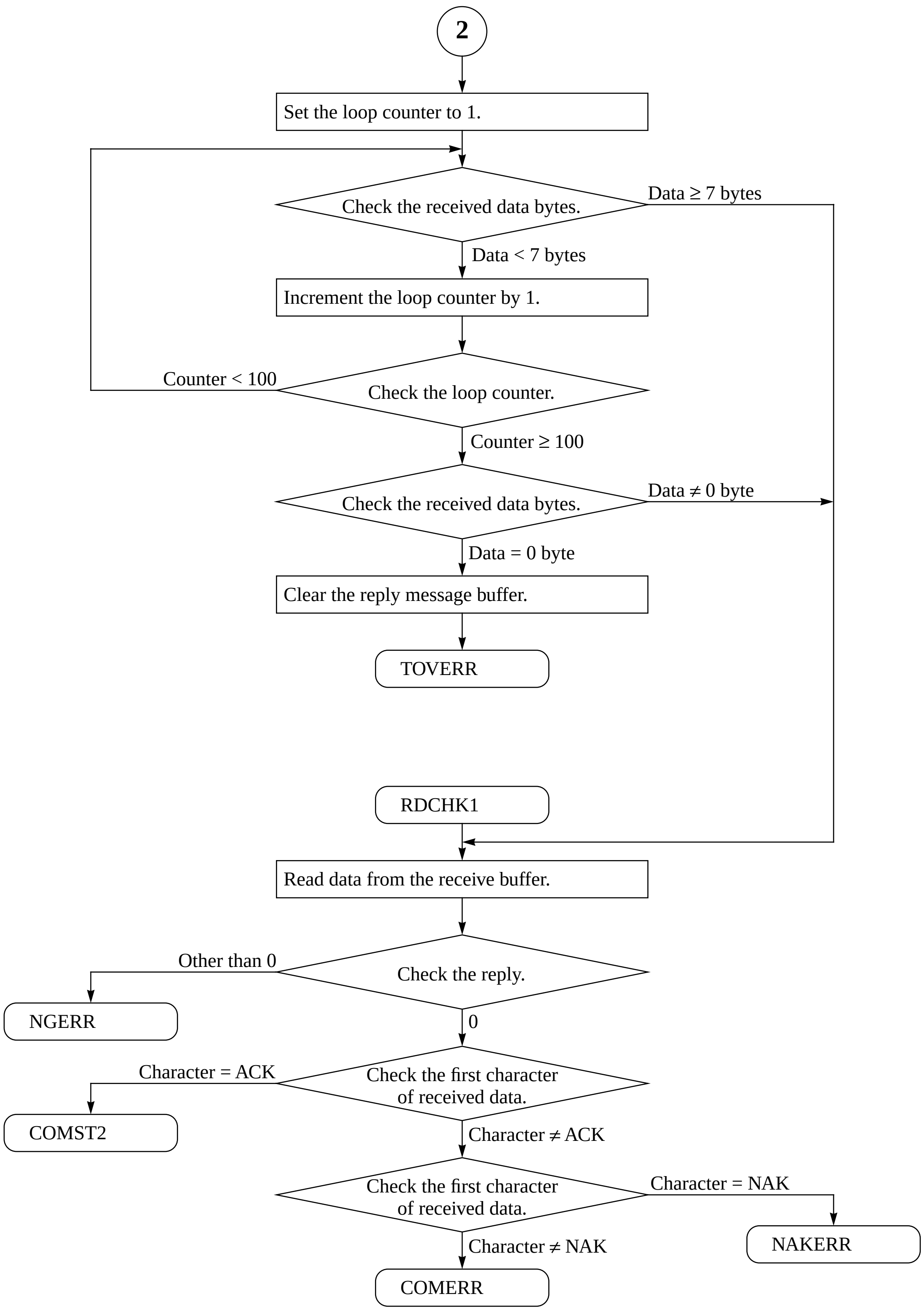
Write User Program from Computer to MICRO³

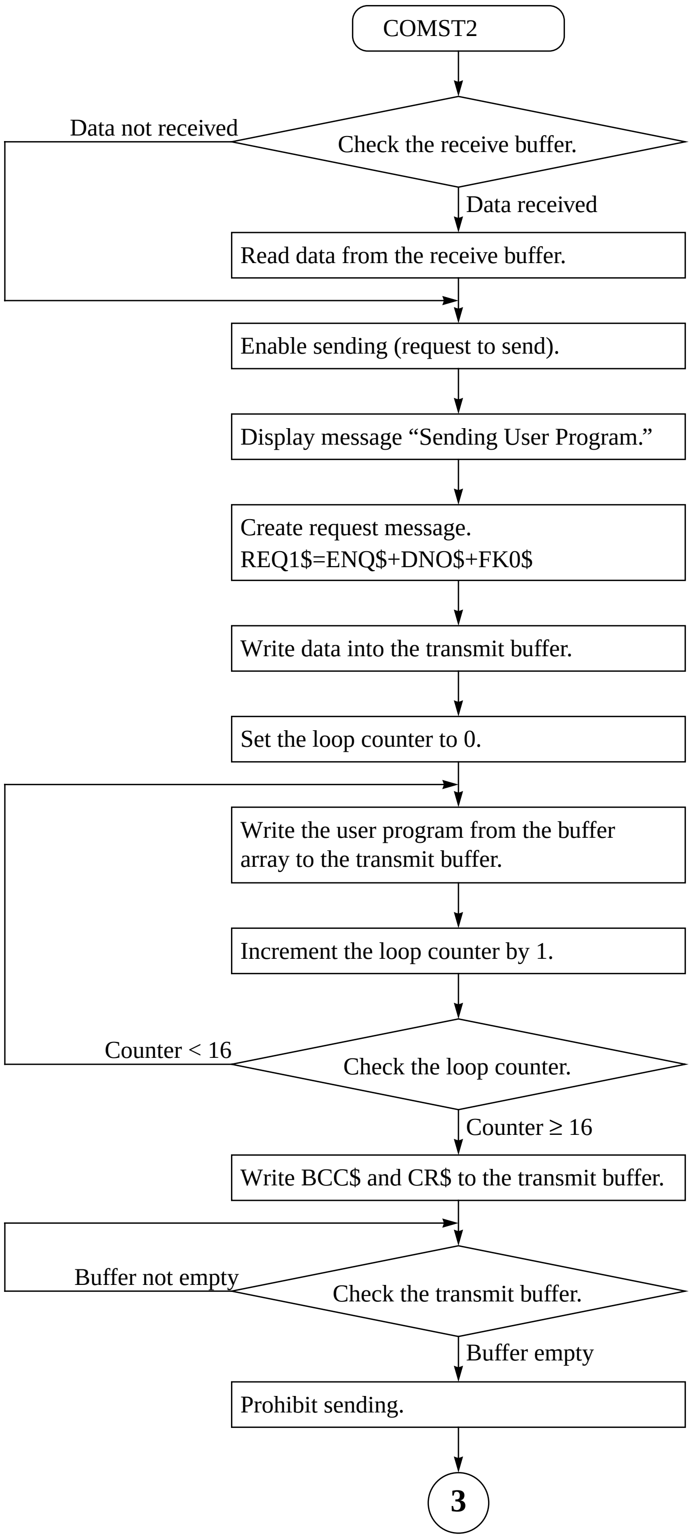
This example demonstrates a program to send a 1K-step user program (filename PROG1) stored on a diskette in drive A to MICRO³ of device number 0. Writing user program from a computer is possible only while MICRO³ is stopped.

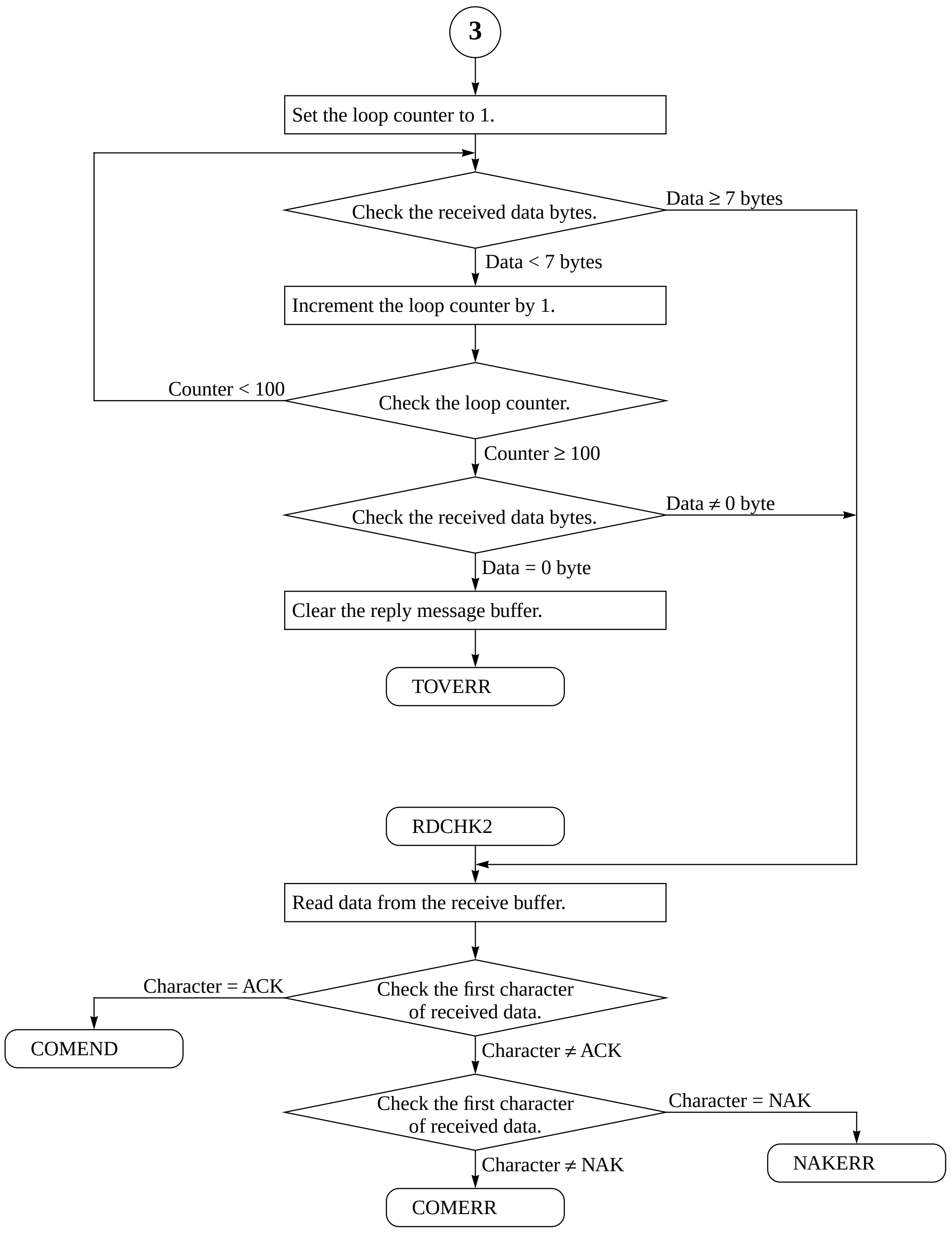
Flow Chart for Writing User Program

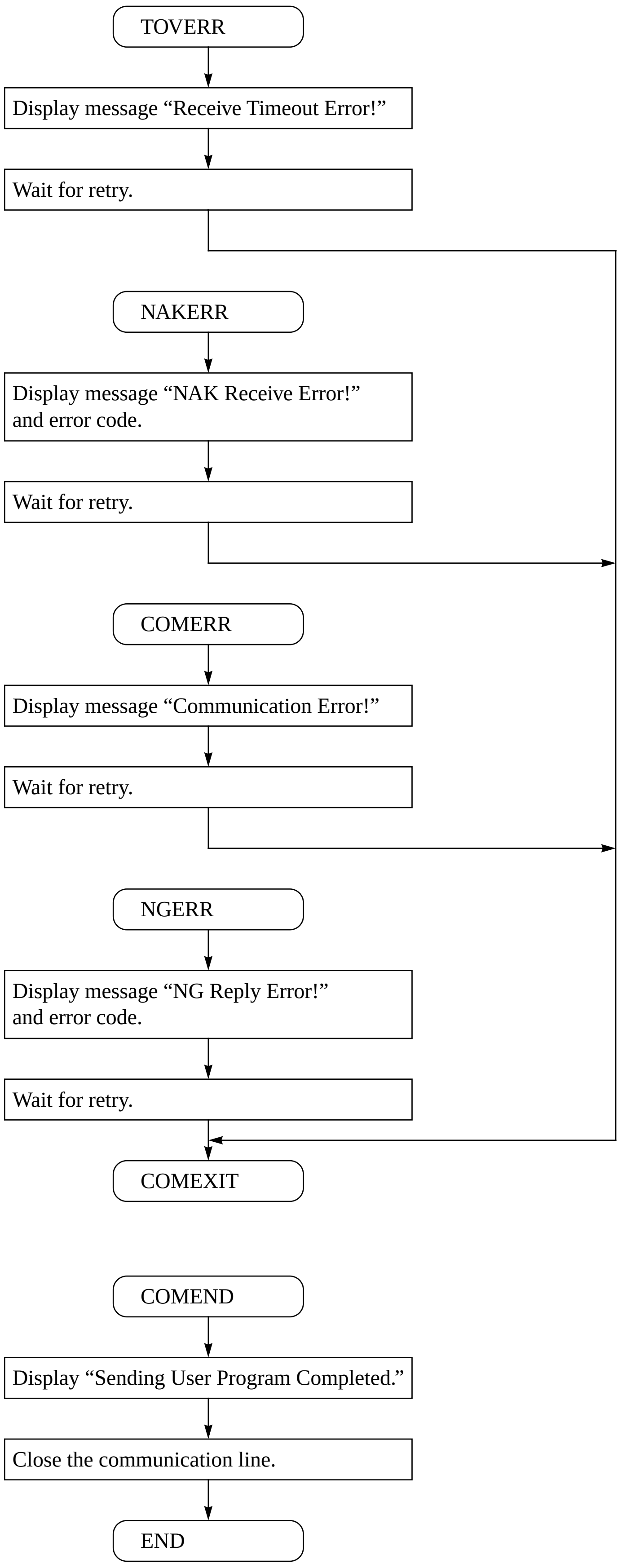

Write 1K-step User Program
Clear the screen.
Specify the maximum value of arrays.
Open the communication line.
(Even parity, 7 data bits, 1 stop bit)
Set control characters.
(ENQ$, ACK$, NAK$, CR$)
Set transmit data.
(DNO$, FK0$, FK1$, CND$, DTK$, SIZ$)
Open the user program file (PROG1).
Display message "Reading User Program from Disk."
Read the user program from the file and store the user program in the buffer array.
Close the user program file (PROG1).
Display message "Calculating BCC."
Create request message.
REQ$=ENQ$+DNO$+FK1$+CND$+DTK$+SIZ$
Calculate BCC. (BCCCAL1)
Set the request message to the buffer for the first frame.
Create request message.
REQ$=ENQ$+DNO$+FK0$
Clear the BCC calculation buffer to 0.
Calculate BCC. (BCCCAL2)
1

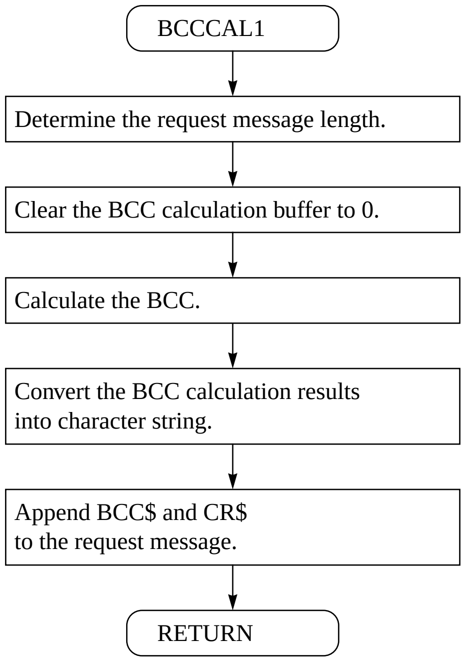

1
Set the loop counter to 0.
Read user program from the buffer array.
Calculate BCC. (BCCCAL2)
Increment the loop counter by 1.
Counter < 16
Check the loop counter.
Counter ≥ 16
Convert the BCC calculation results into character string.
COMST1
Data not received
Check the receive buffer.
Data received
Read data from the receive buffer.
Enable sending (request to send).
Write data into the transmit buffer.
Buffer not empty
Check the transmit buffer.
Buffer empty
Prohibit sending.
2

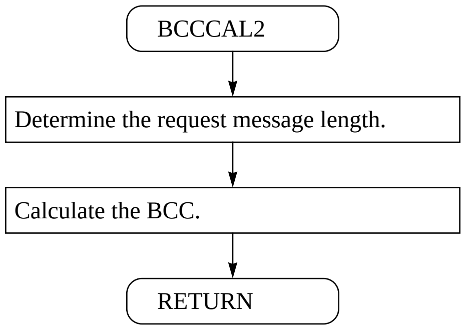

2
Set the loop counter to 1.
Check the received data bytes.
Data ≥ 7 bytes
Data < 7 bytes
Increment the loop counter by 1.
Counter < 100
Check the loop counter.
Counter ≥ 100
Check the received data bytes.
Data ≠ 0 byte
Data = 0 byte
Clear the reply message buffer.
TOVERR
RDCHK1
Read data from the receive buffer.
Other than 0
Check the reply.
NGERR
0
Character = ACK
Check the first character of received data.
COMST2
Character ≠ ACK
Check the first character of received data.
Character = NAK
NAKERR
Character ≠ NAK
COMERR

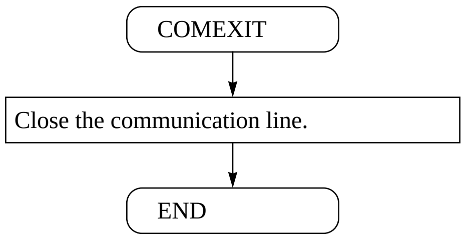

COMST2
Data not received
Check the receive buffer.
Data received
Read data from the receive buffer.
Enable sending (request to send).
Display message "Sending User Program."
Create request message.
REQ1$=ENQ$+DNO$+FK0$
Write data into the transmit buffer.
Set the loop counter to 0.
Write the user program from the buffer array to the transmit buffer.
Increment the loop counter by 1.
Counter < 16
Check the loop counter.
Counter ≥ 16
Write BCC$ and CR$ to the transmit buffer.
Buffer not empty
Check the transmit buffer.
Buffer empty
Prohibit sending.
3


3
Set the loop counter to 1.
Check the received data bytes.
Data ≥ 7 bytes
Data < 7 bytes
Increment the loop counter by 1.
Counter < 100
Check the loop counter.
Counter ≥ 100
Check the received data bytes.
Data ≠ 0 byte
Data = 0 byte
Clear the reply message buffer.
TOVERR
RDCHK2
Read data from the receive buffer.
Character = ACK
Check the first character of received data.
COMEND
Character ≠ ACK
Check the first character of received data.
Character = NAK
NAKERR
Character ≠ NAK
COMERR


TOVERR
Display message “Receive Timeout Error!”
Wait for retry.
NAKERR
Display message “NAK Receive Error!” and error code.
Wait for retry.
COMERR
Display message “Communication Error!”
Wait for retry.
NGERR
Display message “NG Reply Error!” and error code.
Wait for retry.
COMEXIT
COMEND
Display “Sending User Program Completed.”
Close the communication line.
END


BCCCAL1
Determine the request message length.
Clear the BCC calculation buffer to 0.
Calculate the BCC.
Convert the BCC calculation results into character string.
Append BCC$ and CR$ to the request message.
RETURN


BCCCAL2
Determine the request message length.
Calculate the BCC.
RETURN


COMEXIT
Close the communication line.
END

Program List (Filename: WTPROG.BAS)

When executing this sample program, set the MICRO^3 communication format to the default values and the computer communication baud rate to 9600 bps.

```
1000 ' SAVE "WTPROG.BAS",A
1010 '+--------------------------------------------+
1020 '|                                            |
1030 '|        Write User Program                  |
1040 '|                                            |
1050 '|        1K-step User Program                |
1060 '|                                            |
1070 '+--------------------------------------------+
1080 '
10000 CLS
10010 DIM TDT$(17)
10020 '---------------------------------------- [Open Communication Line]
10030 OPEN "COM1:9600,E,7,1" AS #1            'Even parity, 7 data bits, 1 stop bit
10040 '---------------------------------------- [Set Control Characters]
10050 ENQ$=CHR$(&H5)                          'Enquiry
10060 ACK$=CHR$(&H6)                          'Acknowledge
10070 NAK$=CHR$(&H15)                         'Negative acknowledge
10080 CR$ =CHR$(&HD)                          'Terminator code
10090 '---------------------------------------- [Set Transmit Data]
10100 DNO$="00"                               'Device number (00)
10110 FK0$="0"                                'Continuation (Discontinued)
10120 FK1$="1"                                'Continuation (Continued)
10130 CND$="W"                                'Command (Write data)
10140 DTK$="P"                                'Data type (User program)
10150 SIZ$="07FA"                             'Program capacity (1K steps)
10160 '---------------------------------------- [File Control]
10170 OPEN "A:\PROG1" FOR INPUT AS #2         'Open the user program file
10180 PRINT:PRINT "    Reading User Program from Disk   ";
10190 FOR I=0 TO 16
10200   PRINT ".";
10210   INPUT #2,TDT$(I)                      'Read user program from disk file
10220 NEXT I
10230 CLOSE #2                                'Close the user program file
10240 '---------------------------------------- [Create Request Message]
10250 PRINT:PRINT "    Calculating BCC                 ";
10260 REQ$=ENQ$+DNO$+FK1$+CND$+DTK$+SIZ$      'Request message 1 BCC calculation range
10270 GOSUB 11250   :REQ0$=REQ$               'Calculate request message 1 BCC
10280 REQ$=ENQ$+DNO$+FK0$                     'Request message 2 BCC calculation range
10290 BCC=0
10300 GOSUB 11340                             'Calculate request message 2 BCC
10310 FOR J=0 TO 16
10320   PRINT ".";
10330   REQ$=TDT$(J)
10340   GOSUB 11340
10350 NEXT J
10360 BCC$=RIGHT$("0"+HEX$(BCC),2)            'Convert BCC results into character string
10370 '---------------------------------------- [Send Request Message 1]
10380 'COMST1
10390 IF LOC(1)<>0 THEN DUM$=INPUT$(LOC(1),#1)  'Clear receive buffer if not empty
10400 OUT &H3FC,( INP(&H3FC) OR 2 )           'Enable to send
10410 PRINT #1,REQ0$;                         'Write into transmit buffer
10420 IF INP(&H3FD)=&H60 THEN 10430 ELSE 10420  'Wait until transmit buffer is empty
10430 OUT &H3FC,( INP(&H3FC) AND 253 )        'Prohibit to send
10440 '---------------------------------------- [Receive Reply Message 1]
10450 FOR WAITA=1 TO 500                      'Wait for receive data
10460   IF LOC(1)>=7 THEN GOTO 10520          'Check received data bytes
10470 NEXT WAITA
10480 IF LOC(1)<>0 THEN GOTO *RDCHK1          'Jump if receive buffer is not empty
10490 REP$=""                                 'Clear the reply message
10500 GOTO 10920                              'Display timeout error
10510 '---------------------------------------- [Check Reply Message 1]
10520 'RDCHK1
10530 REPLY$=INPUT$(LOC(1),#1)                'Read from receive buffer
10540 IF MID$(REPLY$,4,1)<>"0" THEN GOTO 11150  'Check for NG reply
10550 IF LEFT$(REPLY$,1)=ACK$ THEN GOTO 10590   'Check the first character of received data
```

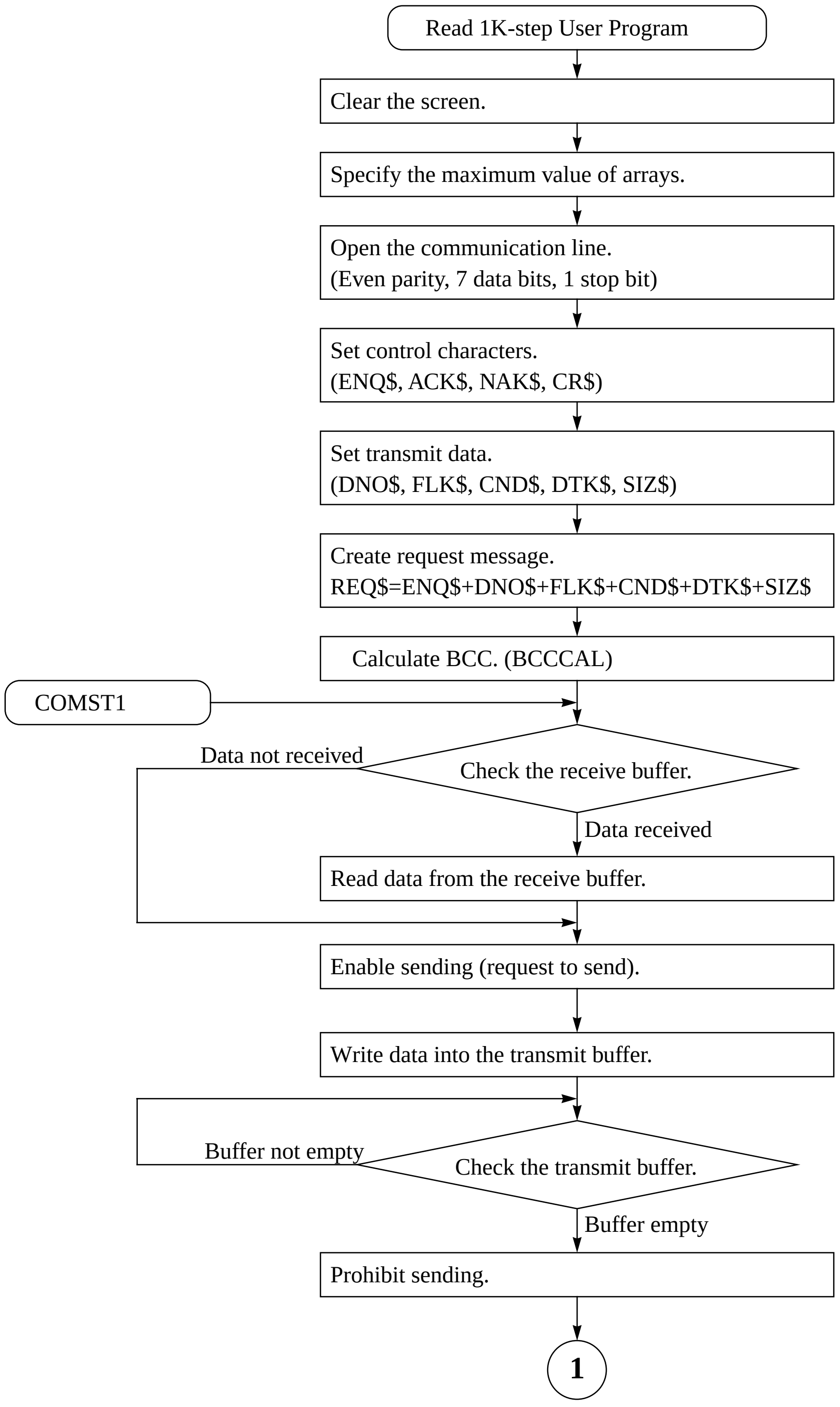
```
10560 IF LEFT$(REPLY$,1)=NAK$ THEN GOTO 10990
10570 GOTO 11080
10580 '----------------------------------------- [Send Request Message 2]
10590 'COMST2
10600 IF LOC(1)<>0 THEN DUM$=INPUT$(LOC(1),#1)  'Clear receive buffer if not empty
10610 OUT &H3FC,( INP(&H3FC) OR 2 )             'Enable to send
10620 PRINT:PRINT "    Sending User Program             ";
10630 REQ1$=ENQ$+DNO$+FK0$                      'BCC calculation range
10640 PRINT #1,REQ1$;                           'Send ENQ, Device No., and Continuation
10650 FOR I=0 TO 16
10660   PRINT ".";
10670   PRINT #1,TDT$(I);                       'Send user program
10680 NEXT I
10690 REQ1$=BCC$+CR$                            'Send BCC and CR
10700 PRINT #1,REQ1$;
10710 IF INP(&H3FD)=&H60 THEN 10720 ELSE 10710  'Wait until transmit buffer is empty
10720 OUT &H3FC,( INP(&H3FC) AND 253 )          'Prohibit to send
10730 '----------------------------------------- [Receive Reply Message 2]
10740 FOR WAITA=1 TO 100                        'Wait for receive data
10750   IF LOC(1)>=7 THEN GOTO 10820            'Check received data bytes
10760 NEXT WAITA
10770 IF LOC(1)<>0 THEN GOTO 10820              'Jump if receive buffer is not empty
10780 REP$=""                                   'Clear the reply message
10790 GOTO 10920                                'Display timeout error
10800 '----------------------------------------- [Check Reply Message 2]
10810 'RDCHK2
10820 REPLY$=INPUT$(LOC(1),#1)                  'Read from receive buffer
10830 IF LEFT$(REPLY$,1)=ACK$ THEN GOTO 10870   'Check the first character of received data
10840 IF LEFT$(REPLY$,1)=NAK$ THEN GOTO 10990
10850 GOTO 11080
10860 '----------------------------------------- [End Communication]
10870 'COMEND
10880 PRINT:PRINT "    Sending User Program Completed   ";
10890 CLOSE #1                                  'Close communication line
10900 END
10910 '----------------------------------------- [Display Timeout Error]
10920 'TOVERR
10930 CLS : BEEP
10940 LOCATE 10,10
10950 PRINT "    Receive Timeout Error !    "
10960 FOR WAITB=1 TO 5000 : NEXT WAITB
10970 GOTO 11410
10980 '----------------------------------------- [Display NAK Receive Error]
10990 'NAKERR
11000 CLS : BEEP
11010 LOCATE 10,10
11020 PRINT "    NAK Receive Error !   "
11030 LOCATE 14,12
11040 PRINT "ERROR CODE=";MID$(REPLY$,5,2)
11050 FOR WAITC=1 TO 5000 : NEXT WAITC
11060 GOTO 11410
11070 '----------------------------------------- [Display Communication Error]
11080 'COMERR
11090 CLS : BEEP
11100 LOCATE 10,10
11110 PRINT "    Communication Error !    "
11120 FOR WAITD=1 TO 5000 : NEXT WAITD
11130 GOTO 11410
11140 '----------------------------------------- [Display NG Reply Error]
11150 'NGERR
11160 CLS : BEEP
11170 LOCATE 10,10
11180 PRINT "   NG Reply Error !    "
11190 LOCATE 14,12
11200 PRINT "ERROR CODE=";MID$(REPLY$,5,2)
11210 FOR WAITE=1 TO 5000 : NEXT WAITE
11220 GOTO 11410
11230 '----------------------------------------- [BCC Calculation 1]
11240 'BCCCAL1
11250 REQL=LEN(REQ$)                            'Determine BCC calculation range
11260 BCC=0                                     'Initialize BCC
```

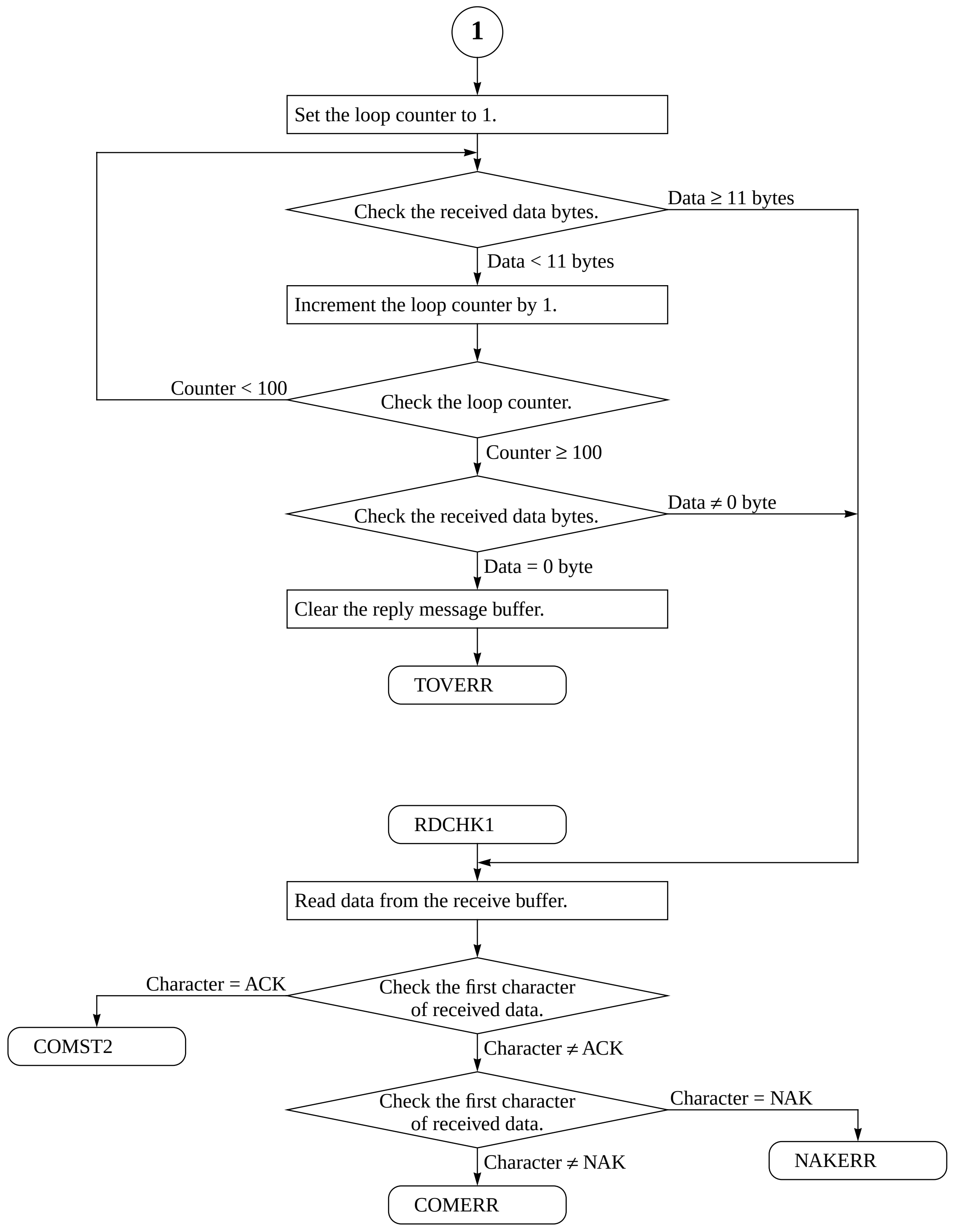
```
11270 FOR I=1 TO REQL                          'Repeat BCC calculation range
11280   BCC=BCC XOR ASC(MID$(REQ$,I,1))        'Calculate BCC
11290 NEXT I
11300 BCC$=RIGHT$("0"+HEX$(BCC),2)             'Convert BCC results into character string
11310 REQ$=REQ$+BCC$+CR$                       'Append BCC and CR to request message
11320 RETURN
11330 '----------------------------------------- [BCC Calculation 2]
11340 'BCCCAL2
11350 REQL=LEN(REQ$)                           'Determine BCC calculation range
11360 FOR I=1 TO REQL                          'Repeat BCC calculation range
11370   BCC=BCC XOR ASC(MID$(REQ$,I,1))        'Calculate BCC
11380 NEXT I
11390 RETURN
11400 '----------------------------------------- [Exit Communication]
11410 'COMEXIT
11420 CLOSE #1                                 'Close communication line
11430 END
```

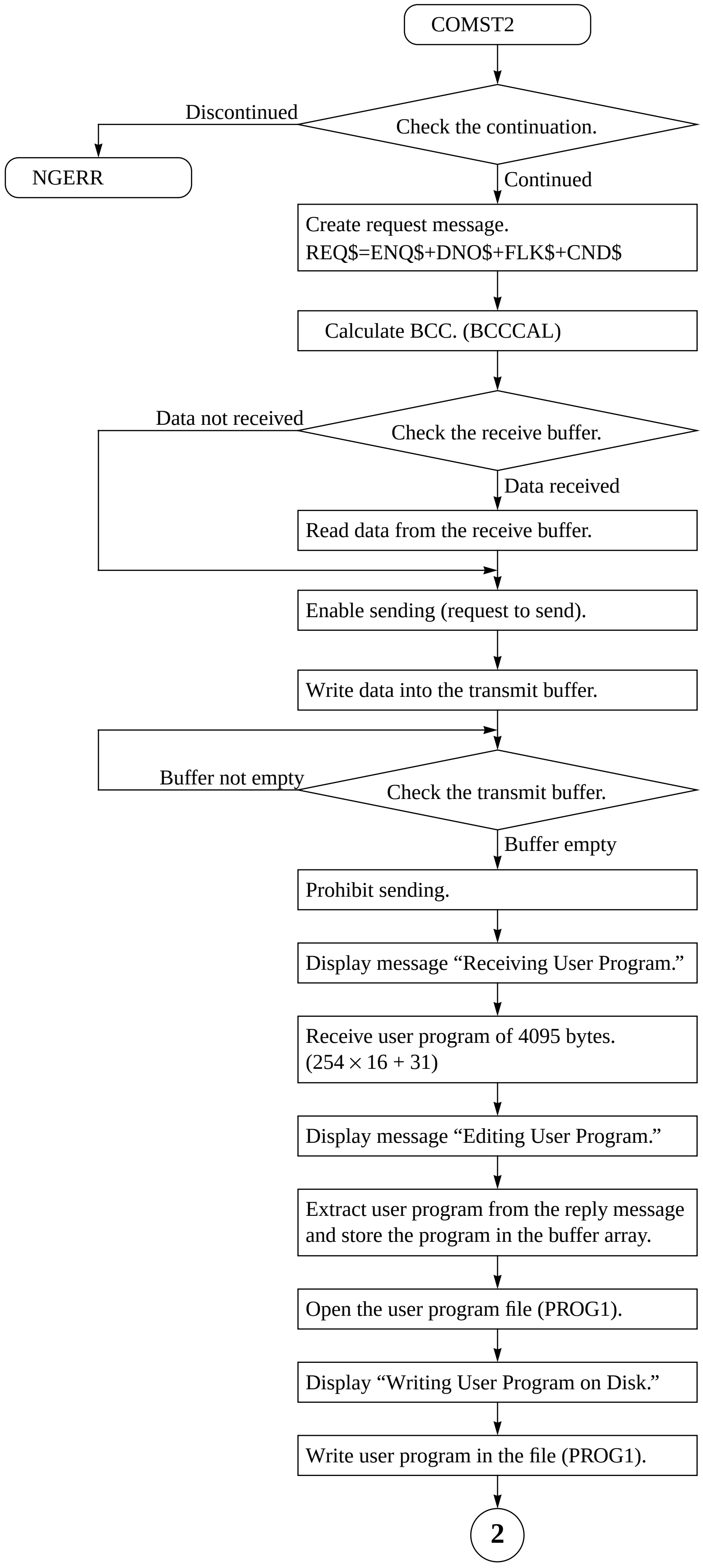
Read User Program from MICRO^3 to Computer

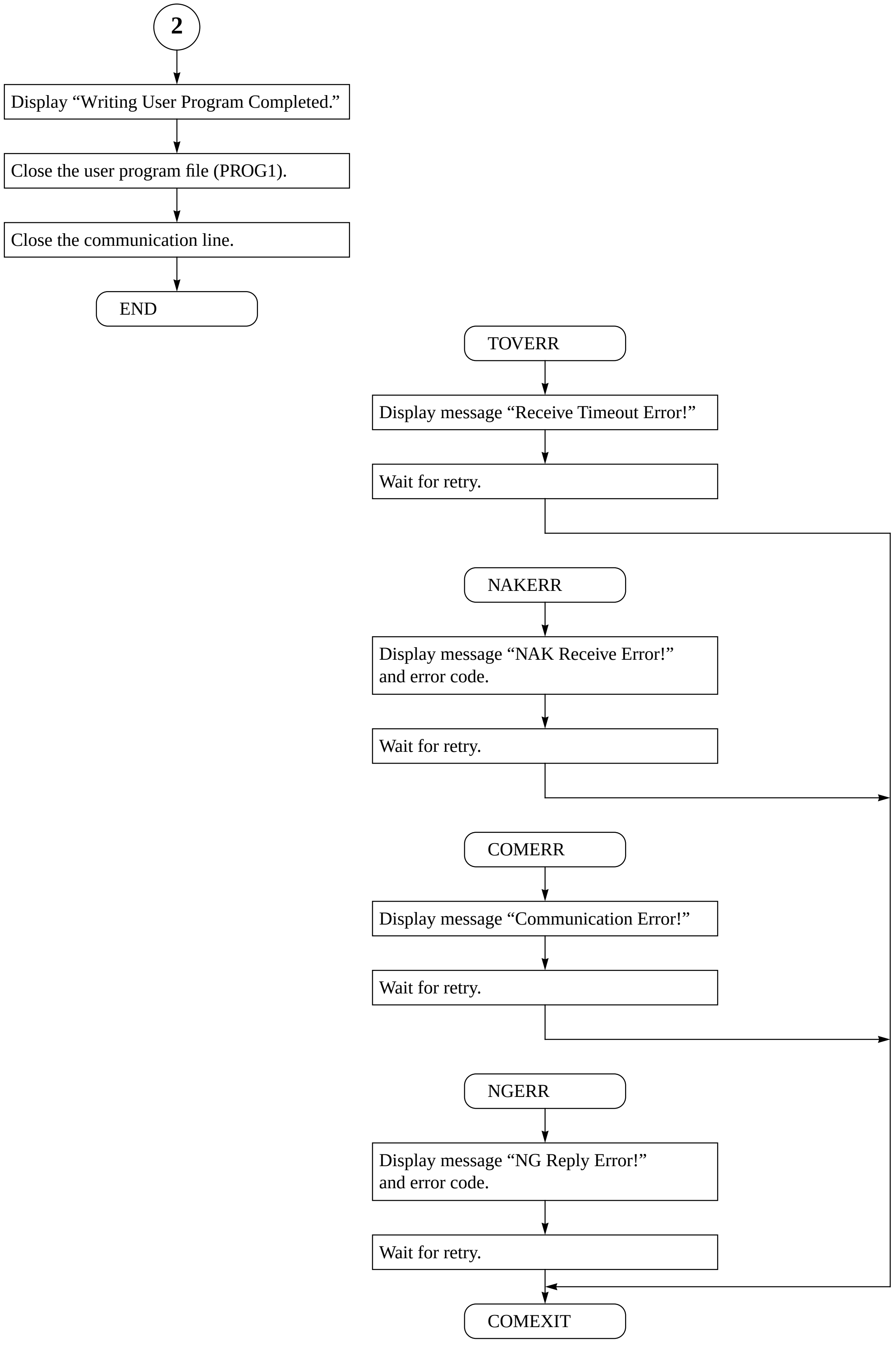
This example demonstrates a program to read a 1K-step user program from the MICRO^3 of device number 0 to the computer and store the user program (filename PROG1) on a diskette in drive A. Reading user program from MICRO^3 is possible whether MICRO^3 is running or stopped.

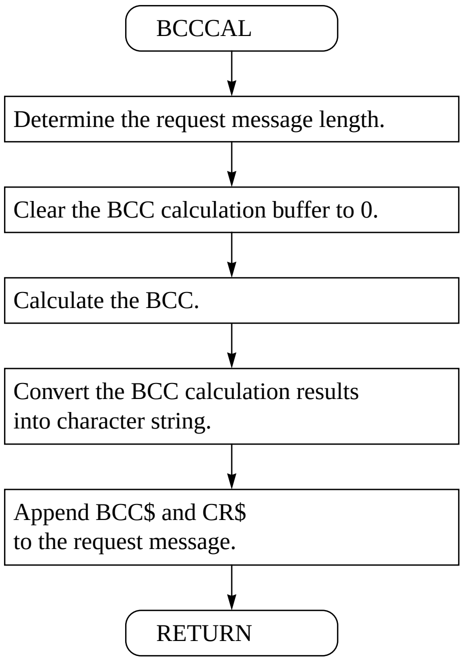
Flow Chart for Reading User Program

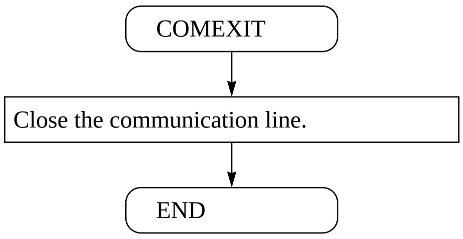

Read 1K-step User Program
Clear the screen.
Specify the maximum value of arrays.
Open the communication line. (Even parity, 7 data bits, 1 stop bit)
Set control characters. (ENQ$, ACK$, NAK$, CR$)
Set transmit data. (DNO$, FLK$, CND$, DTK$, SIZ$)
Create request message. REQ$=ENQ$+DNO$+FLK$+CND$+DTK$+SIZ$
Calculate BCC. (BCCCAL)
COMST1
Data not received
Check the receive buffer.
Data received
Read data from the receive buffer.
Enable sending (request to send).
Write data into the transmit buffer.
Buffer not empty
Check the transmit buffer.
Buffer empty
Prohibit sending.
1


1
Set the loop counter to 1.
Check the received data bytes.
Data ≥ 11 bytes
Data < 11 bytes
Increment the loop counter by 1.
Counter < 100
Check the loop counter.
Counter ≥ 100
Check the received data bytes.
Data ≠ 0 byte
Data = 0 byte
Clear the reply message buffer.
TOVERR
RDCHK1
Read data from the receive buffer.
Character = ACK
Check the first character of received data.
COMST2
Character ≠ ACK
Check the first character of received data.
Character = NAK
NAKERR
Character ≠ NAK
COMERR


COMST2
Check the continuation.
Discontinued
NGERR
Continued
Create request message.
REQ$=ENQ$+DNO$+FLK$+CND$
Calculate BCC. (BCCCAL)
Check the receive buffer.
Data not received
Data received
Read data from the receive buffer.
Enable sending (request to send).
Write data into the transmit buffer.
Check the transmit buffer.
Buffer not empty
Buffer empty
Prohibit sending.
Display message "Receiving User Program."
Receive user program of 4095 bytes.
(254 × 16 + 31)
Display message "Editing User Program."
Extract user program from the reply message and store the program in the buffer array.
Open the user program file (PROG1).
Display "Writing User Program on Disk."
Write user program in the file (PROG1).
2


2
Display "Writing User Program Completed."
Close the user program file (PROG1).
Close the communication line.
END
TOVERR
Display message "Receive Timeout Error!"
Wait for retry.
NAKERR
Display message "NAK Receive Error!" and error code.
Wait for retry.
COMERR
Display message "Communication Error!"
Wait for retry.
NGERR
Display message "NG Reply Error!" and error code.
Wait for retry.
COMEXIT


BCCCAL
Determine the request message length.
Clear the BCC calculation buffer to 0.
Calculate the BCC.
Convert the BCC calculation results into character string.
Append BCC$ and CR$ to the request message.
RETURN


COMEXIT
Close the communication line.
END

Program List (Filename: RDPROG.BAS)

When executing this sample program, set the MICRO3 communication format to the default values and the computer communication baud rate to 9600 bps.

```
1000 'SAVE "RDPROG.BAS",A
1010 '+---------------------------------------------+
1020 '|                                             |
1030 '|          Read User Program                  |
1040 '|                                             |
1050 '|          1K-step User Program               |
1060 '|                                             |
1070 '+---------------------------------------------+
1080 '
10000 CLS
10010 DIM REPLY$(17):DIM REP(2048)
10020 '------------------------------------------ [Open Communication Line]
10030 OPEN "COM1:9600,E,7,1" AS #1              'Even parity, 7 data bits, 1 stop bit
10040 '------------------------------------------ [Set Control Characters]
10050 ENQ$=CHR$(&H5)                            'Enquiry
10060 ACK$=CHR$(&H6)                            'Acknowledge
10070 NAK$=CHR$(&H15)                           'Negative acknowledge
10080 CR$ =CHR$(&HD)                            'Terminator code
10090 '------------------------------------------ [Set Transmit Data]
10100 DNO$="00"                                 'Device number (00)
10110 FLK$="0"                                  'Continuation (Discontinued)
10120 CND$="R"                                  'Command (Read data)
10130 DTK$="P"                                  'Data type (User program)
10140 SIZ$="07FA"                               'Program capacity (1K steps)
10150 '------------------------------------------ [Create Request Message 1]
10160 REQ$=ENQ$+DNO$+FLK$+CND$+DTK$+SIZ$        'Request message 1 BCC calculation range
10170 GOSUB 11210                               'Calculate request message 1 BCC
10180 '------------------------------------------ [Send Request Message 1]
10190 'COMST1
10200 IF LOC(1)<>0 THEN DUM$=INPUT$(LOC(1),#1)  'Clear receive buffer if not empty
10210 OUT &H3FC,( INP(&H3FC) OR 2 )             'Enable to send
10220 PRINT #1,REQ$;                            'Write into transmit buffer
10230 IF INP(&H3FD)=&H60 THEN 10240 ELSE 10230  'Wait until transmit buffer is empty
10240 OUT &H3FC,( INP(&H3FC) AND 253 )          'Prohibit to send
10250 '------------------------------------------ [Receive Reply Message 1]
10260 FOR WAITA=1 TO 100                        'Wait for receive data
10270   IF LOC(1)>=11 THEN 10330                'Check received data bytes
10280 NEXT WAITA
10290 IF LOC(1)<>0 THEN GOTO 10330              'Jump if receive buffer is not empty
10300 REP$=""                                   'Clear the reply message
10310 GOTO 10890                                'Display timeout error
10320 '------------------------------------------ [Check Reply Message 1]
10330 'RDCHK1
10340 REPLY$=INPUT$(LOC(1),#1)                  'Read from receive buffer
10350 IF LEFT$(REPLY$,1)=ACK$ THEN GOTO 10390   'Check the first character of received data
10360 IF LEFT$(REPLY$,1)=NAK$ THEN GOTO 10960
10370 GOTO 11050
10380 '------------------------------------------ [Create Request Message 2]
10390 'COMST2
10400 IF MID$(REPLY$,4,1)<>"1" THEN GOTO 11120
10410 REQ$=ENQ$+DNO$+FLK$+CND$                  'Request message 2 BCC calculation range
10420 GOSUB 11210                               'Calculate request message 2 BCC
10430 '------------------------------------------ [Send Request Message 2]
10440 IF LOC(1)<>0 THEN DUM$=INPUT$(LOC(1),#1)  'Clear receive buffer if not empty
10450 OUT &H3FC,( INP(&H3FC) OR 2 )             'Enable to send
10460 PRINT #1,REQ$;                            'Write into transmit buffer
10470 IF INP(&H3FD)=&H60 THEN 10480 ELSE 10470  'Wait until transmit buffer is empty
10480 OUT &H3FC,( INP(&H3FC) AND 253 )          'Prohibit to send
10490 '------------------------------------------ [Receive Reply Message 2]
10500 'RDCHK2
10510 PRINT:PRINT "    Receiving User Program         ";
10520 FOR I=0 TO 15                             'Read every 254 bytes
10530   PRINT ".";
10540   REPLY$(I)=INPUT$(254,#1)                'Read from receive buffer
10550 NEXT I
```

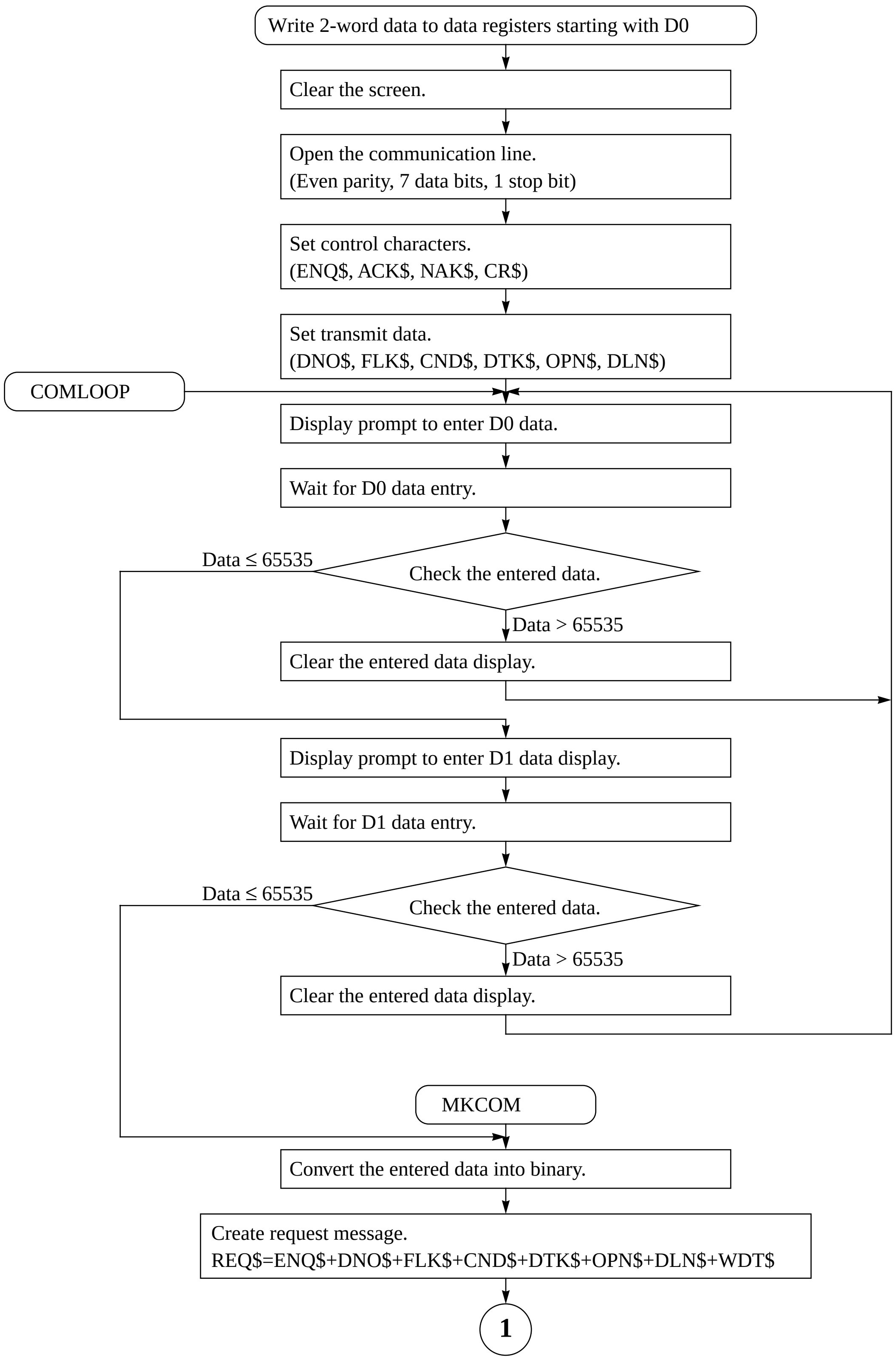
```
10560 REPLY$(16)=INPUT$(31,#1)                  'Read from receive buffer
10570 WCT=0
10580 '----------------------------------------- [Copy Data]
10590 PRINT:PRINT "    Editing User Program           ";
10600 FOR J=5 TO 254 STEP 2                     'Eliminate unnecessary data and copy to buffer
10610   REP(WCT)=VAL("&H"+MID$(REPLY$(0),J,2))
10620   WCT=WCT+1
10630 NEXT J
10640   PRINT ".";
10650 FOR L=1 TO 15                             'Copy all data to buffer
10660   FOR M=1 TO 254 STEP 2
10670     REP(WCT)=VAL("&H"+MID$(REPLY$(L),M,2))
10680     WCT=WCT+1
10690   NEXT M
10700   PRINT ".";
10710 NEXT L
10720 FOR N=1 TO 28 STEP 2                      'Copy remaining data to buffer
10730   REP(WCT)=VAL("&H"+MID$(REPLY$(16),N,2))
10740   WCT=WCT+1
10750 NEXT N:
10760 '----------------------------------------- [File Control]
10770 OPEN "A:PROG1" FOR OUTPUT AS #2           'Open the user program file
10780 PRINT:PRINT "    Writing User Program on Disk   ";
10790 WSC=0
10800 FOR I=0 TO 2043                           'Write user program in disk file
10810   PRINT #2,RIGHT$("0"+HEX$(REP(I)),2);
10820   WSC=WSC+1:IF WSC=125 THEN WSC=0:PRINT ".";
10830 NEXT I
10840 PRINT:PRINT "    Writing User Program Completed    "
10850 CLOSE #2                                  'Close the user program file
10860 CLOSE #1                                  'Close communication line
10870 END
10880 '----------------------------------------- [Display Timeout Error]
10890 'TOVERR
10900 CLS : BEEP
10910 LOCATE 10,10
10920 PRINT "    Receive Timeout Error !    "
10930 FOR WAITB=1 TO 5000 : NEXT WAITB
10940 GOTO 11310
10950 '----------------------------------------- [Display NAK Receive Error]
10960 'NAKERR
10970 CLS : BEEP
10980 LOCATE 10,10
10990 PRINT "    NAK Receive Error !   "
11000 LOCATE 14,12
11010 PRINT "ERROR CODE=";MID$(REPLY$,5,2)
11020 FOR WAITC=1 TO 5000 : NEXT WAITC
11030 GOTO 11310
11040 '----------------------------------------- [Display Communication Error]
11050 'COMERR
11060 CLS : BEEP
11070 LOCATE 10,10
11080 PRINT "    Communication Error !    "
11090 FOR WAITD=1 TO 5000 : NEXT WAITD
11100 GOTO 11310
11110 '----------------------------------------- [Display NG Reply Error]
11120 'NGERR
11130 CLS : BEEP
11140 LOCATE 10,10
11150 PRINT "   NG Reply Error !    "
11160 LOCATE 14,12
11170 PRINT "ERROR CODE=";MID$(REPLY$,5,2)
11180 FOR WAITE=1 TO 5000 : NEXT WAITE
11190 GOTO 11310
11200 '----------------------------------------- [BCC Calculation]
11210 'BCCCAL
11220 REQL=LEN(REQ$)                            'Determine BCC calculation range
11230 BCC=0                                     'Initialize BCC
11240 FOR I=1 TO REQL                           'Repeat BCC calculation range
11250   BCC=BCC XOR ASC(MID$(REQ$,I,1))         'Calculate BCC
11260 NEXT I
```

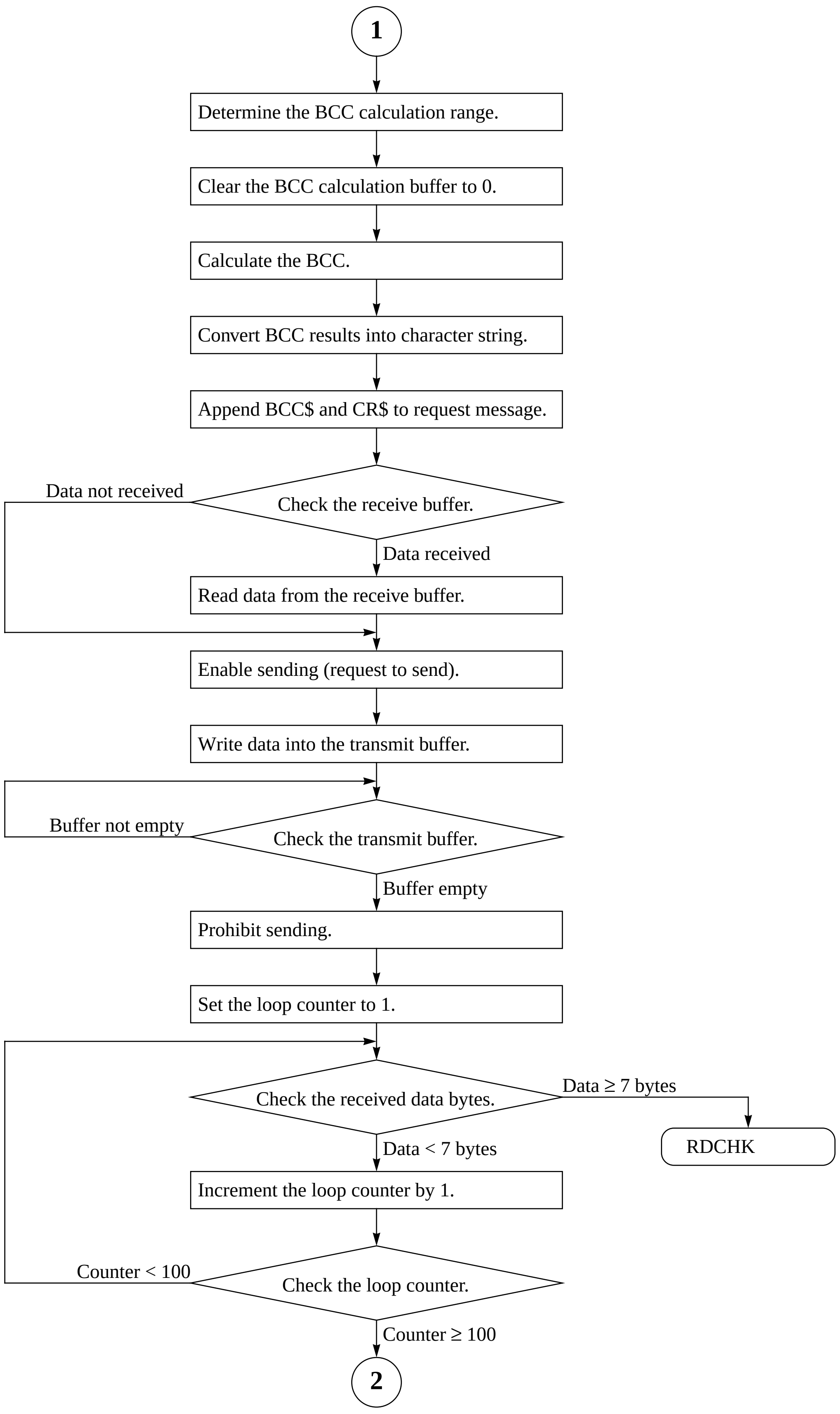
```
11270 BCC$=RIGHT$("0"+HEX$(BCC),2)              'Convert BCC results into character string
11280 REQ$=REQ$+BCC$+CR$                        'Append BCC and CR to request message
11290 RETURN
11300 '----------------------------------------- [Exit Communication]
11310 'COMEXIT
11320 CLOSE #1                                  'Close communication line
11330 END
```

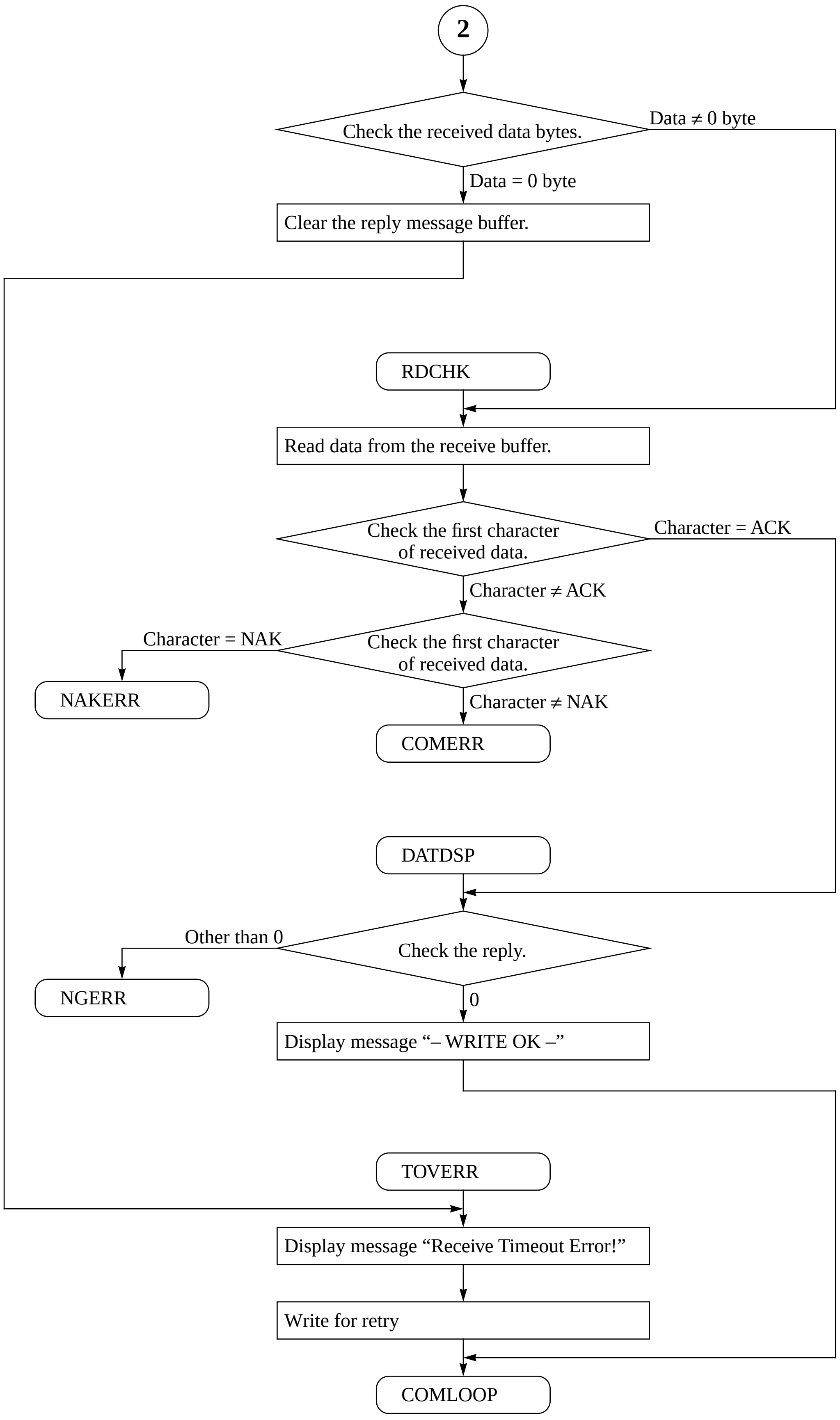
Write N Bytes to Data Registers

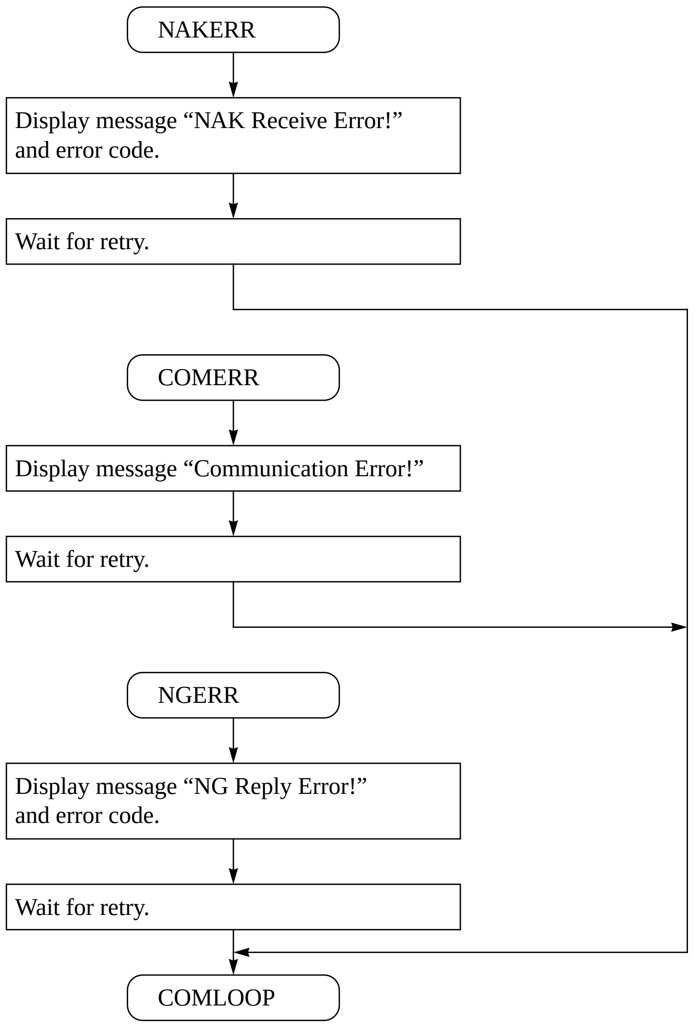
This example demonstrates a program to write two numeric values entered from the keyboard to data registers D0 and D1 in the MICRO3 of device number 0. Writing data from the computer is possible whether MICRO3 is running or stopped.

Flow Chart for Writing N Bytes to Data Registers


Write 2-word data to data registers starting with D0
Clear the screen.
Open the communication line.
(Even parity, 7 data bits, 1 stop bit)
Set control characters.
(ENQ$, ACK$, NAK$, CR$)
Set transmit data.
(DNO$, FLK$, CND$, DTK$, OPN$, DLN$)
COMLOOP
Display prompt to enter D0 data.
Wait for D0 data entry.
Data ≤ 65535
Check the entered data.
Data > 65535
Clear the entered data display.
Display prompt to enter D1 data display.
Wait for D1 data entry.
Data ≤ 65535
Check the entered data.
Data > 65535
Clear the entered data display.
MKCOM
Convert the entered data into binary.
Create request message.
REQ$=ENQ$+DNO$+FLK$+CND$+DTK$+OPN$+DLN$+WDT$
1


1
Determine the BCC calculation range.
Clear the BCC calculation buffer to 0.
Calculate the BCC.
Convert BCC results into character string.
Append BCC$ and CR$ to request message.
Data not received
Check the receive buffer.
Data received
Read data from the receive buffer.
Enable sending (request to send).
Write data into the transmit buffer.
Buffer not empty
Check the transmit buffer.
Buffer empty
Prohibit sending.
Set the loop counter to 1.
Data ≥ 7 bytes
Check the received data bytes.
Data < 7 bytes
RDCHK
Increment the loop counter by 1.
Counter < 100
Check the loop counter.
Counter ≥ 100
2


2
Check the received data bytes.
Data ≠ 0 byte
Data = 0 byte
Clear the reply message buffer.
RDCHK
Read data from the receive buffer.
Check the first character of received data.
Character = ACK
Character ≠ ACK
Character = NAK
Check the first character of received data.
NAKERR
Character ≠ NAK
COMERR
DATDSP
Other than 0
Check the reply.
NGERR
0
Display message "– WRITE OK –"
TOVERR
Display message "Receive Timeout Error!"
Write for retry
COMLOOP


NAKERR
Display message "NAK Receive Error!" and error code.
Wait for retry.
COMERR
Display message "Communication Error!"
Wait for retry.
NGERR
Display message "NG Reply Error!" and error code.
Wait for retry.
COMLOOP

Program List (Filename: WTNBYT.BAS)

When executing this sample program, set the MICRO^3 communication format to the default values and the computer communication baud rate to 9600 bps.

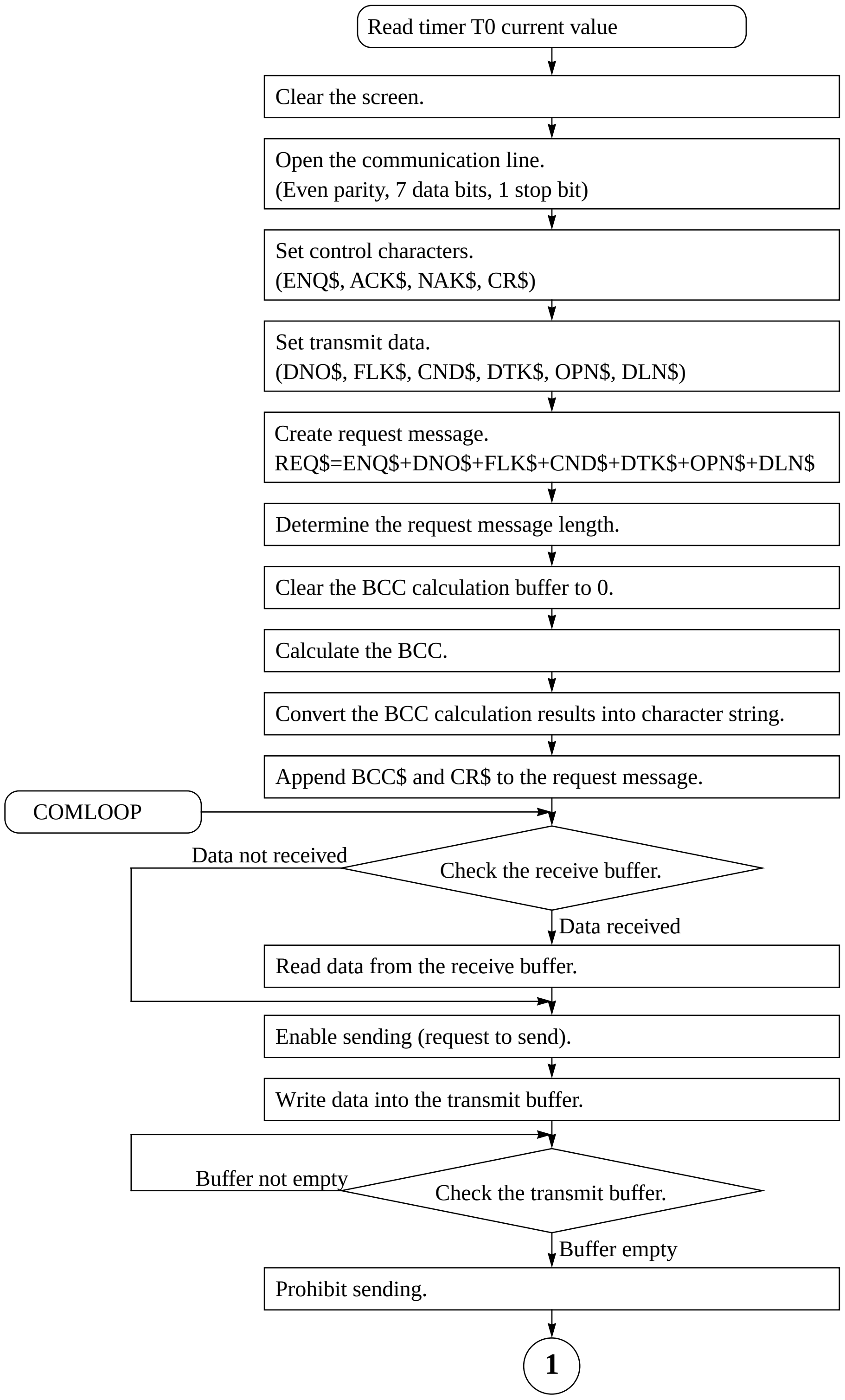
```
1000 'SAVE "WTNBYT.BAS",A
1010 '+-----------------------------------------------------+
1020 '|                                                     |
1030 '|        Write N Bytes                                |
1040 '|                                                     |
1050 '|        2 words (4 bytes) starting with Data Register D0  |
1060 '|                                                     |
1070 '+-----------------------------------------------------+
1080 '
10000 CLS
10010 '----------------------------------------- [Open Communication Line]
10020 OPEN "COM1:9600,E,7,1" AS #1             'Even parity, 7 data bits, 1 stop bit
10030 '----------------------------------------- [Set Control Characters]
10040 ENQ$=CHR$(&H5)                           'Enquiry
10050 ACK$=CHR$(&H6)                           'Acknowledge
10060 NAK$=CHR$(&H15)                          'Negative acknowledge
10070 CR$ =CHR$(&HD)                           'Terminator code
10080 '----------------------------------------- [Set Transmit Data]
10090 DNO$="00"                                'Device number (00)
10100 FLK$="0"                                 'Continuation (Discontinued)
10110 CND$="W"                                 'Command (Write data)
10120 DTK$="D"                                 'Data type (Data register)
10130 OPN$="0000"                              'Operand number (0000)
10140 DLN$="04"                                'Data length (4 bytes)
10150 '----------------------------------------- [Create Request Message]
10160 'COMLOOP
10170 O$=OPN$
10180 O$=STR$(VAL(O$)+1)
10190 O$=RIGHT$("000"+RIGHT$(O$,LEN(O$)-1),4)
10200 LOCATE 10,6
10210 PRINT " INPUT ";DTK$;OPN$;" DATA : ";
10220 INPUT "",WDT0$                           'Input data to D0
10230 IF VAL(WDT0$) <= 65535! THEN GOTO 10260   'Check the inputted data
10240 LOCATE 10,26:PRINT "          "
10250 BEEP:GOTO 10200
10260 LOCATE 11,6
10270 PRINT " INPUT ";DTK$;O$;" DATA : ";
10280 INPUT "",WDT1$                           'Input data to D1
10290 IF VAL(WDT1$) <= 65535! THEN GOTO 10320   'Check the inputted data
10300 LOCATE 11,26:PRINT "          "
10310 BEEP:GOTO 10260
10320 'MKCOM
10330 WD0$=WDT0$:WD1$=WDT1$
10340 WDT0$=RIGHT$("000"+HEX$(VAL(WDT0$)),4)    'Convert BCD to binary
10350 WDT1$=RIGHT$("000"+HEX$(VAL(WDT1$)),4)
10360 WDT$=WDT0$+WDT1$
10370 REQ$=ENQ$+DNO$+FLK$+CND$+DTK$+OPN$+DLN$+WDT$ 'BCC calculation range
10380 REQL=LEN(REQ$)                           'Determine BCC calculation range
10390 BCC=0                                    'Initialize BCC
10400 FOR I=1 TO REQL                          'Repeat BCC calculation range
10410   BCC=BCC XOR ASC(MID$(REQ$,I,1))        'Calculate BCC
10420 NEXT I
10430 BCC$=RIGHT$("0"+HEX$(BCC),2)             'Convert BCC results into character string
10440 REQ$=REQ$+BCC$+CR$                       'Append BCC and CR to request message
10450 '----------------------------------------- [Send Request Message]
10460 IF LOC(1)<>0 THEN DUM$=INPUT$(LOC(1),#1)  'Clear receive buffer if not empty
10470 OUT &H3FC,( INP(&H3FC) OR 2 )            'Enable to send
10480 PRINT #1,REQ$;                           'Write into transmit buffer
10490 IF INP(&H3FD)=&H60 THEN 10500 ELSE 10490  'Wait until transmit buffer is empty
10500 OUT &H3FC,( INP(&H3FC) AND 253 )         'Prohibit to send
10510 '----------------------------------------- [Receive Reply Message]
10520 FOR WAITA=1 TO 100                       'Wait for receive data
10530   IF LOC(1)>=7 THEN GOTO 10590           'Check received data bytes
10540 NEXT WAITA
10550 IF LOC(1)<>0 THEN GOTO 10590             'Jump if receive buffer is not empty
```

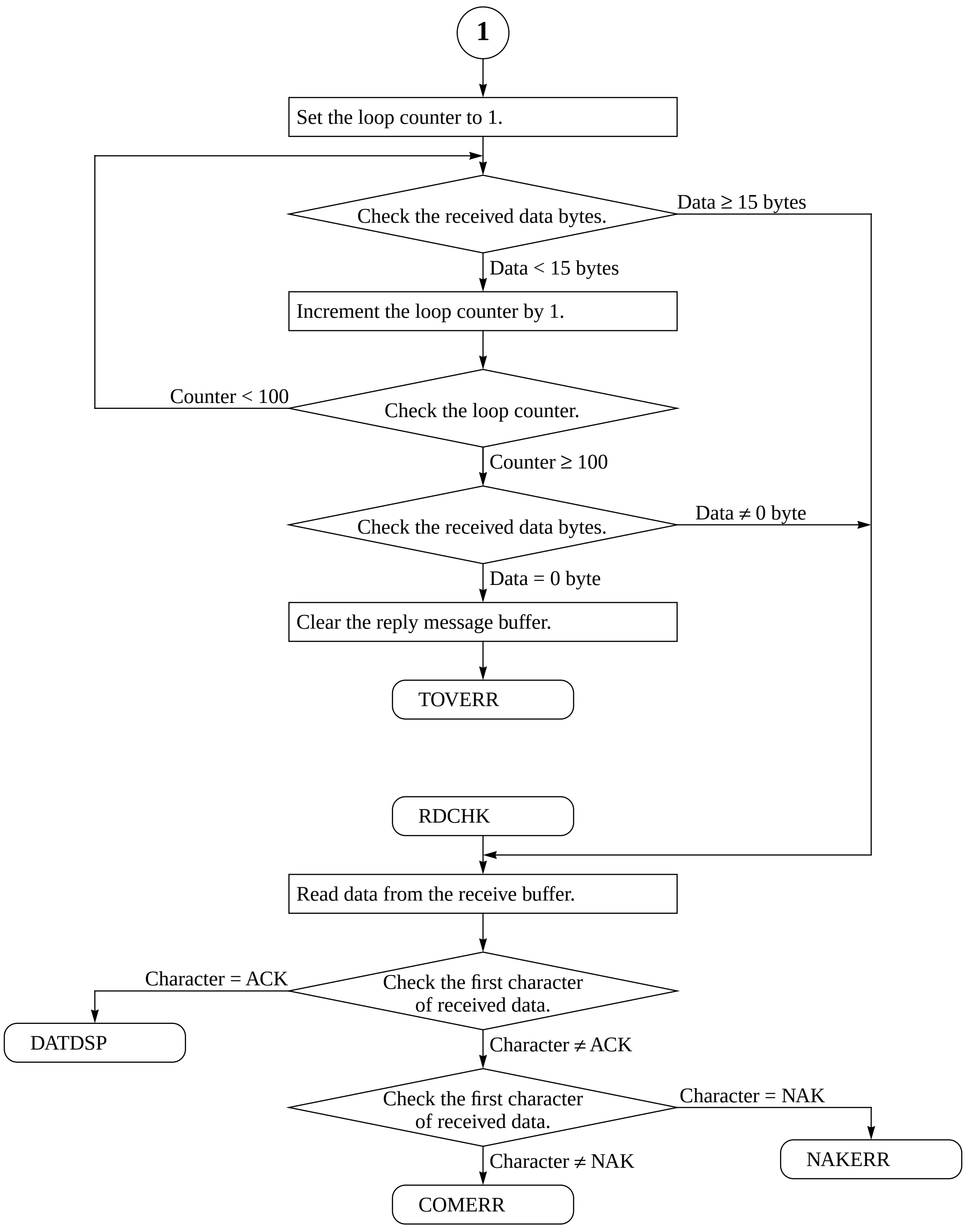
```
10560 REP$=""                                   'Clear the reply message
10570 GOTO 10770                                'Display timeout error
10580 '------------------------------------------ [Check Reply Message]
10590 'RDCHK
10600 REPLY$=INPUT$(LOC(1),#1)                  'Read from receive buffer
10610 IF LEFT$(REPLY$,1)=ACK$ THEN GOTO 10650   'Check the first character of received data
10620 IF LEFT$(REPLY$,1)=NAK$ THEN GOTO 10840
10630 GOTO 10930
10640 '------------------------------------------ [Display DR Writing Results]
10650 'DATDSP
10660 IF MID$(REPLY$,4,1)<>"0" THEN GOTO 11000
10670 LOCATE 12,10
10680 PRINT "                                               "
10690 LOCATE 12,10
10700 PRINT DTK$;OPN$;" <- ";WD0$;" , ";
10710 PRINT DTK$;O$;" <- ";WD1$;
10720 PRINT ST$; "    - WRITE OK -"
10730 LOCATE 10,26:PRINT "          "
10740 LOCATE 11,26:PRINT "          "
10750 GOTO 10160
10760 '------------------------------------------ [Display Timeout Error]
10770 'TOVERR
10780 CLS : BEEP
10790 LOCATE 10,10
10800 PRINT "    Receive Timeout Error !    "
10810 FOR WAITB=1 TO 5000 : NEXT WAITB
10820 CLS : GOTO 10160
10830 '------------------------------------------ [Display NAK Receive Error]
10840 'NAKERR
10850 CLS : BEEP
10860 LOCATE 10,10
10870 PRINT "    NAK Receive Error !   "
10880 LOCATE 14,12
10890 PRINT "ERROR CODE=";MID$(REPLY$,5,2)
10900 FOR WAITC=1 TO 5000 : NEXT WAITC
10910 CLS : GOTO 10160
10920 '------------------------------------------ [Display Communication Error]
10930 'COMERR
10940 CLS : BEEP
10950 LOCATE 10,10
10960 PRINT "    Communication Error !    "
10970 FOR WAITD=1 TO 5000 : NEXT WAITD
10980 CLS : GOTO 10160
10990 '------------------------------------------ [Display NG Reply Error]
11000 'NGERR
11010 CLS : BEEP
11020 LOCATE 10,10
11030 PRINT "   NG Reply Error !    "
11040 LOCATE 14,12
11050 PRINT "ERROR CODE=";MID$(REPLY$,5,2)
11060 FOR WAITE=1 TO 5000 : NEXT WAITE
11070 CLS : GOTO 10160
```

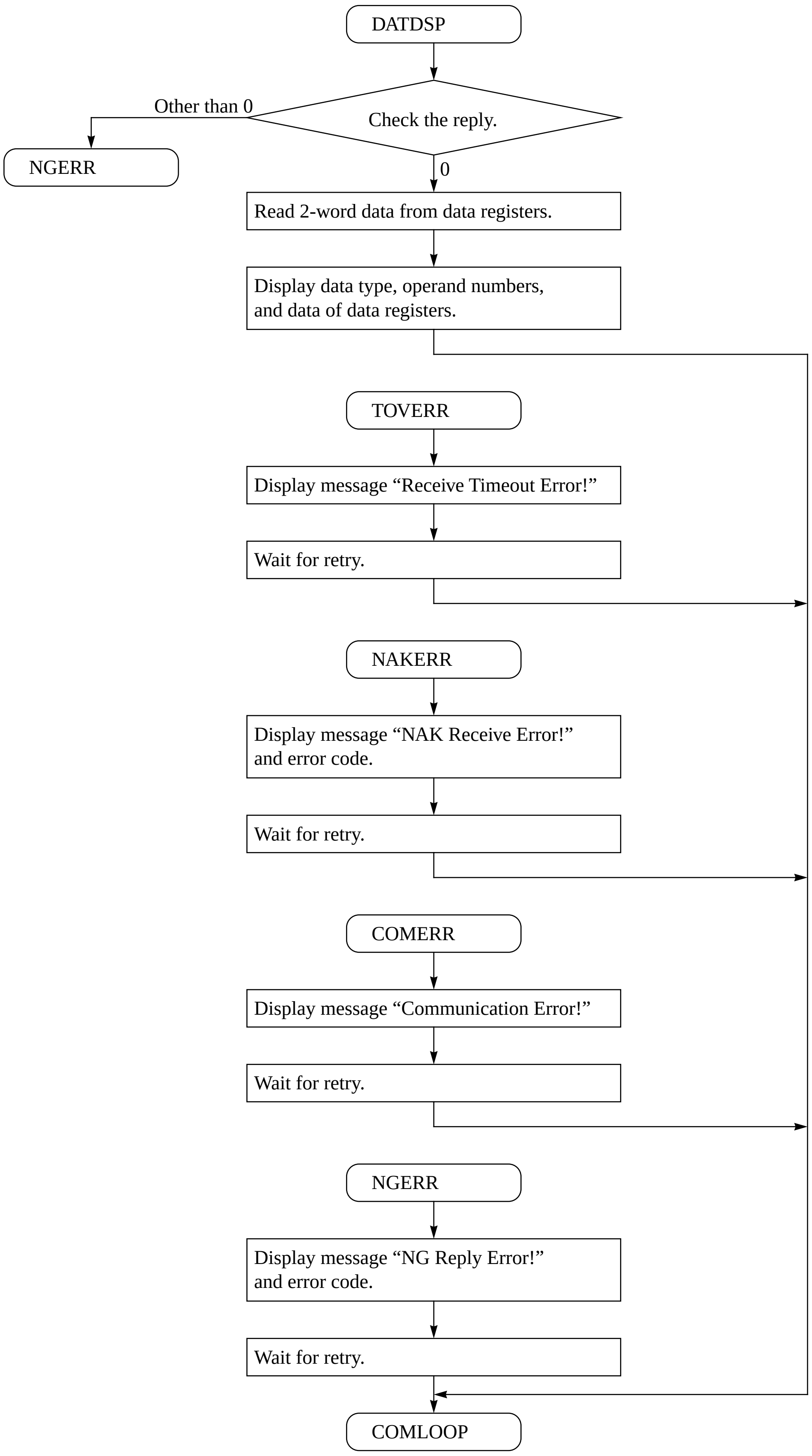
Read N Bytes from Data Registers

This example demonstrates a program to read two numeric values from data registers D0 and D1 in the MICRO^3 of device number 0 and display the values on the computer screen. Reading data from MICRO^3 is possible whether MICRO^3 is running or stopped.

Flow Chart for Reading N Bytes from Data Registers


Read timer T0 current value
Clear the screen.
Open the communication line.
(Even parity, 7 data bits, 1 stop bit)
Set control characters.
(ENQ$, ACK$, NAK$, CR$)
Set transmit data.
(DNO$, FLK$, CND$, DTK$, OPN$, DLN$)
Create request message.
REQ$=ENQ$+DNO$+FLK$+CND$+DTK$+OPN$+DLN$
Determine the request message length.
Clear the BCC calculation buffer to 0.
Calculate the BCC.
Convert the BCC calculation results into character string.
Append BCC$ and CR$ to the request message.
COMLOOP
Data not received
Check the receive buffer.
Data received
Read data from the receive buffer.
Enable sending (request to send).
Write data into the transmit buffer.
Buffer not empty
Check the transmit buffer.
Buffer empty
Prohibit sending.
1


1
Set the loop counter to 1.
Check the received data bytes.
Data ≥ 15 bytes
Data < 15 bytes
Increment the loop counter by 1.
Counter < 100
Check the loop counter.
Counter ≥ 100
Check the received data bytes.
Data ≠ 0 byte
Data = 0 byte
Clear the reply message buffer.
TOVERR
RDCHK
Read data from the receive buffer.
Character = ACK
Check the first character of received data.
DATDSP
Character ≠ ACK
Check the first character of received data.
Character = NAK
NAKERR
Character ≠ NAK
COMERR


DATDSP
Check the reply.
Other than 0
NGERR
0
Read 2-word data from data registers.
Display data type, operand numbers, and data of data registers.
TOVERR
Display message "Receive Timeout Error!"
Wait for retry.
NAKERR
Display message "NAK Receive Error!" and error code.
Wait for retry.
COMERR
Display message "Communication Error!"
Wait for retry.
NGERR
Display message "NG Reply Error!" and error code.
Wait for retry.
COMLOOP

Program List (Filename: RDNBYT.BAS)

When executing this sample program, set the MICRO^3 communication format to the default values and the computer communication baud rate to 9600 bps.

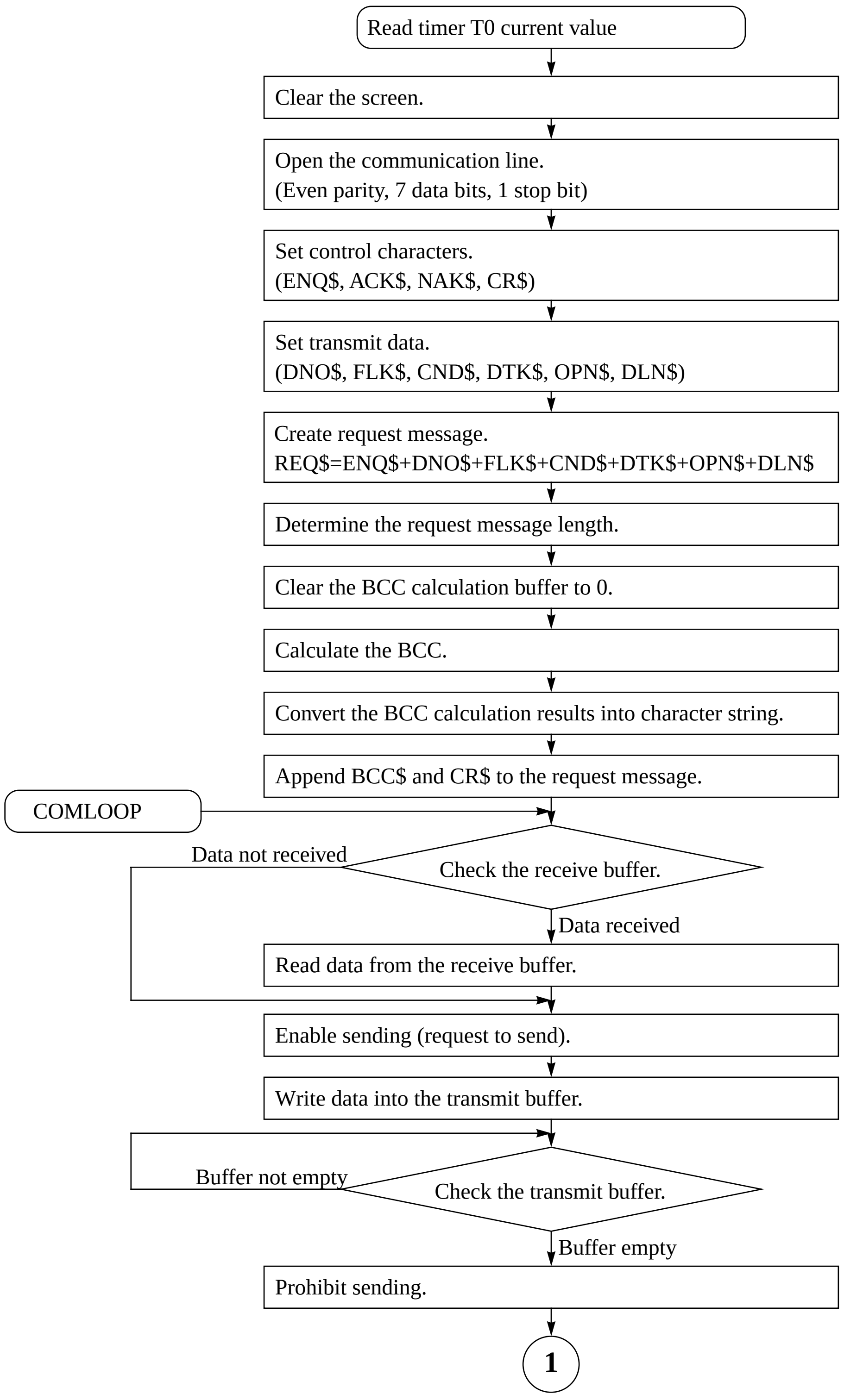
```
1000 'SAVE "RDNBYT.BAS",A
1010 '+-------------------------------------------------------+
1020 '|                                                       |
1030 '|        Read N Bytes                                   |
1040 '|        2 words (4 bytes) starting with Data Register D0  |
1050 '|                                                       |
1060 '|                                                       |
1070 '+-------------------------------------------------------+
1080 '
10000 CLS
10010 '----------------------------------------- [Open Communication Line]
10020 OPEN "COM1:9600,E,7,1" FOR RANDOM AS #1   'Even parity, 7 data bits, 1 stop bit
10030 '----------------------------------------- [Set Control Characters]
10040 ENQ$ = CHR$(&H5)                          'Enquiry
10050 ACK$ = CHR$(&H6)                          'Acknowledge
10060 NAK$ = CHR$(&H15)                         'Negative acknowledge
10070 CR$ = CHR$(&HD)                           'Terminator code
10080 '----------------------------------------- [Set Transmit Data]
10090 DNO$ = "00"                               'Device number (00)
10100 FLK$ = "0"                                'Continuation (Discontinued)
10110 CND$ = "R"                                'Command (Read data)
10120 DTK$ = "D"                                'Data type (Data register)
10130 OPN$ = "0000"                             'Operand number (0000)
10140 DLN$ = "04"                               'Data length (4 bytes)
10150 '----------------------------------------- [Create Request Message]
10160 REQ$=ENQ$+DNO$+FLK$+CND$+DTK$+OPN$+DLN$   'BCC calculation range
10170 REQL = LEN(REQ$)                          'Determine BCC calculation range
10180 BCC = 0                                   'Initialize BCC
10190 FOR I = 1 TO REQL                         'Repeat BCC calculation range
10200   BCC = BCC XOR ASC(MID$(REQ$, I, 1))     'Calculate BCC
10210 NEXT I
10220 BCC$ = RIGHT$("0" + HEX$(BCC), 2)         'Convert BCC results into character string
10230 REQ$ = REQ$ + BCC$ + CR$                  'Append BCC and CR to request message
10240 '----------------------------------------- [Send Request Message]
10250 'COMLOOP
10255 FOR I = 1 TO 100: NEXT
10260 IF LOC(1)<>0 THEN DUM$=INPUT$(LOC(1),#1)  'Clear receive buffer if not empty
10270 OUT &H3FC, (INP(&H3FC) OR 2)              'Enable to send
10280 PRINT #1, REQ$;                           'Write into transmit buffer
10290 IF INP(&H3FD)=&H60 THEN 10300 ELSE 10290  'Wait until transmit buffer is empty
10300 OUT &H3FC, (INP(&H3FC) AND 253)           'Prohibit to send
10310 '----------------------------------------- [Receive Reply Message]
10320 FOR WAITA = 1 TO 1000                     'Wait for receive data
10330   IF LOC(1) >= 15 THEN GOTO 10390         'Check received data bytes
10340 NEXT WAITA
10350 IF LOC(1) <> 0 THEN GOTO 10390            'Jump if receive buffer is not empty
10360 REPLY$ = ""                               'Clear the reply message
10370 GOTO 10620                                'Display timeout error
10380 '----------------------------------------- [Check Reply Message]
10390 'RDCHK
10400 REPLY$ = INPUT$(LOC(1), #1)               'Read from receive buffer
10410 IF LEFT$(REPLY$,1)=ACK$ THEN GOTO 10450   'Check the first character of received data
10420 IF LEFT$(REPLY$, 1) = NAK$ THEN GOTO 10840
10430 GOTO 10780
10440 '----------------------------------------- [Display Data Register]
10450 'DATDSP
10460 IF MID$(REPLY$, 4, 1) <> "0" THEN GOTO 10850
10470 LOCATE 10, 10
10480 O$ = OPN$
10490 FOR J = 1 TO 2                            'Repeat DR display 2 times
10500   DAT!=VAL("&H"+MID$(REPLY$,J*4+1,4))     'Select data of data register
10510   PRINT DTK$; O$; "  ";                   'Display data register number and data
10520   IF DAT! >= 0 THEN GOTO 10550
10530   PRINT USING "#####"; 65536! - ABS(DAT);
10540   GOTO 10560
```

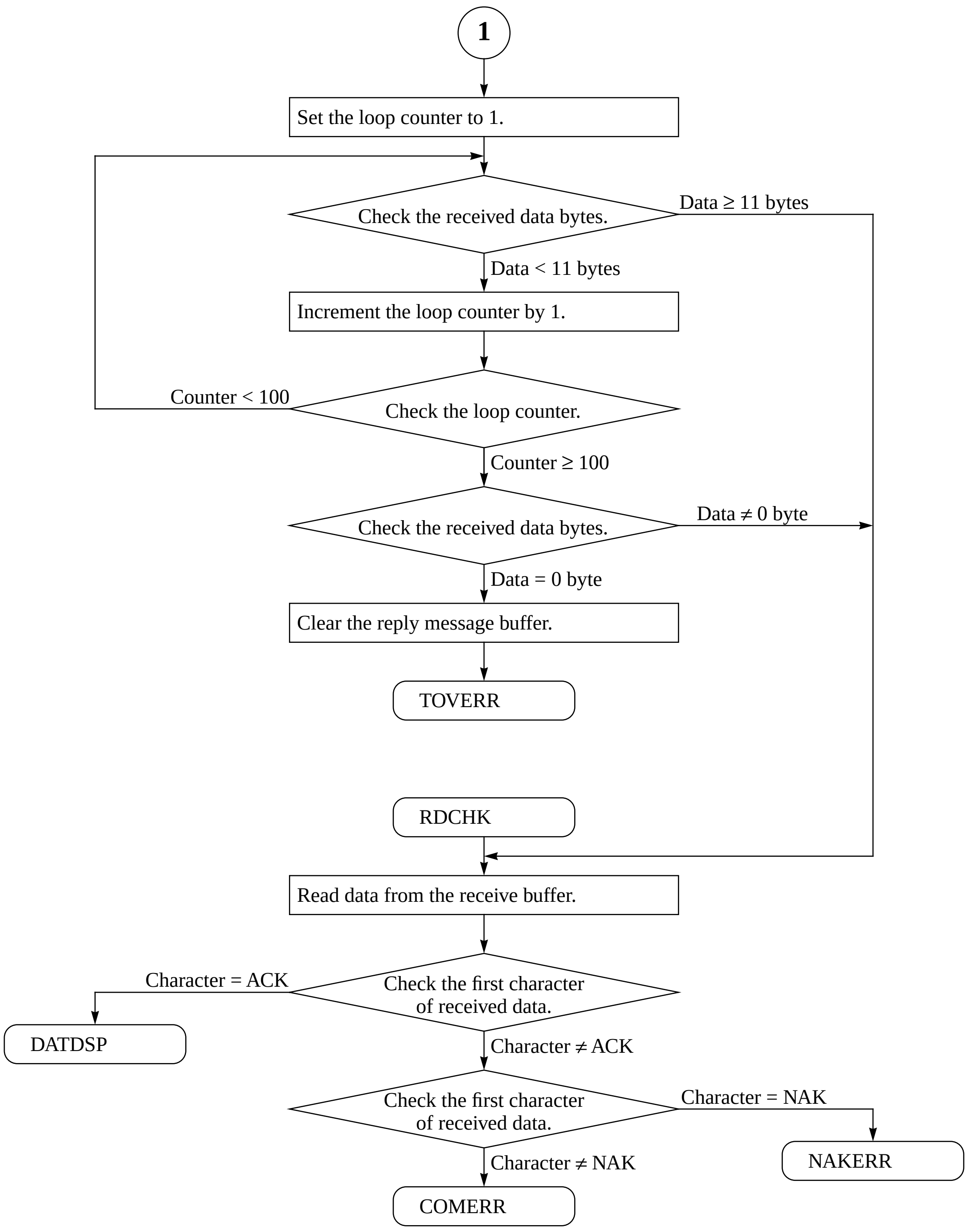
```
10550   PRINT USING "#####"; DAT;
10560   PRINT "     ";
10570   O$ = STR$(VAL(O$) + 1)                  'Increment data register number by 1
10580   O$=RIGHT$("000"+RIGHT$(O$,LEN(O$)-1),4) 'Convert DR number to character string
10590 NEXT J
10600 GOTO 10250
10610 '---------------------------------------- [Display Timeout Error]
10620 'TOVERR
10630 CLS : 'BEEP
10640 LOCATE 10, 10
10650 PRINT "    Receive Timeout Error !    "
10660 FOR WAITB = 1 TO 5000: NEXT WAITB
10665 TOV = TOV + 1
10670 CLS : GOTO 10250
10680 '---------------------------------------- [Display NAK Receive Error]
10690 'NAKERR
10700 CLS : BEEP
10710 LOCATE 10, 10
10720 PRINT "    NAK Receive Error !   "
10730 LOCATE 14, 12
10740 PRINT "ERROR CODE="; MID$(REPLY$, 5, 2)
10750 FOR WAITC = 1 TO 5000: NEXT WAITC
10755 NR = NR + 1
10760 CLS : GOTO 10250
10770 '---------------------------------------- [Display Communication Error
10780 'COMERR
10790 CLS : BEEP
10800 LOCATE 10, 10
10810 PRINT "    Communication Error !   "
10820 FOR WAITD = 1 TO 5000: NEXT WAITD
10825 TR = TR + 1
10830 CLS : GOTO 10250
10840 '---------------------------------------- [Display NG Reply Error]
10850 'NGERR
10860 CLS : BEEP
10870 LOCATE 10, 10
10880 PRINT "   NG Reply Error !    "
10890 LOCATE 14, 12
10900 PRINT "ERROR CODE="; MID$(REPLY$, 5, 2)
10910 FOR WAITE = 1 TO 5000: NEXT WAITE
10920 CLS : GOTO 10250
```

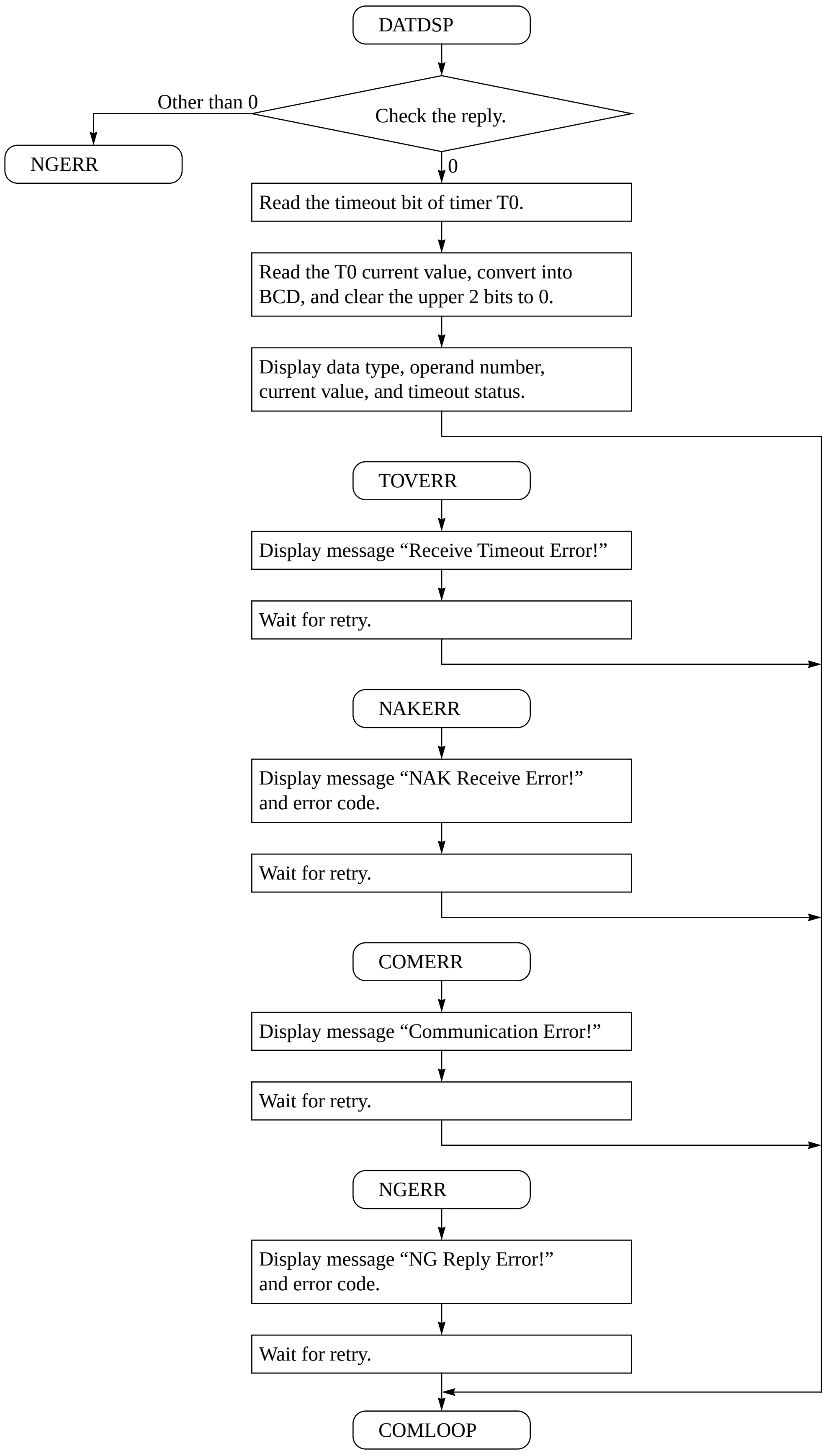
Read N Bytes from Timer / Counter Current Value

This example demonstrates a program to read the current value of timer T0 or counter C0, whichever in the user program, from the MICRO^3 of device number 0 and display the value and timeout/countout status on the computer screen. Reading data from MICRO^3 is possible whether MICRO^3 is running or stopped.

Flow Chart for Reading N Bytes from Timers


Read timer T0 current value
Clear the screen.
Open the communication line.
(Even parity, 7 data bits, 1 stop bit)
Set control characters.
(ENQ$, ACK$, NAK$, CR$)
Set transmit data.
(DNO$, FLK$, CND$, DTK$, OPN$, DLN$)
Create request message.
REQ$=ENQ$+DNO$+FLK$+CND$+DTK$+OPN$+DLN$
Determine the request message length.
Clear the BCC calculation buffer to 0.
Calculate the BCC.
Convert the BCC calculation results into character string.
Append BCC$ and CR$ to the request message.
COMLOOP
Data not received
Check the receive buffer.
Data received
Read data from the receive buffer.
Enable sending (request to send).
Write data into the transmit buffer.
Buffer not empty
Check the transmit buffer.
Buffer empty
Prohibit sending.
1


1
Set the loop counter to 1.
Check the received data bytes.
Data ≥ 11 bytes
Data < 11 bytes
Increment the loop counter by 1.
Counter < 100
Check the loop counter.
Counter ≥ 100
Check the received data bytes.
Data ≠ 0 byte
Data = 0 byte
Clear the reply message buffer.
TOVERR
RDCHK
Read data from the receive buffer.
Character = ACK
Check the first character of received data.
DATDSP
Character ≠ ACK
Check the first character of received data.
Character = NAK
NAKERR
Character ≠ NAK
COMERR


DATDSP
Check the reply.
Other than 0
NGERR
0
Read the timeout bit of timer T0.
Read the T0 current value, convert into BCD, and clear the upper 2 bits to 0.
Display data type, operand number, current value, and timeout status.
TOVERR
Display message "Receive Timeout Error!"
Wait for retry.
NAKERR
Display message "NAK Receive Error!" and error code.
Wait for retry.
COMERR
Display message "Communication Error!"
Wait for retry.
NGERR
Display message "NG Reply Error!" and error code.
Wait for retry.
COMLOOP

Program List (Filename: RDTBYT.BAS)

When executing this sample program, set the MICRO3 communication format to the default values and the computer communication baud rate to 9600 bps.

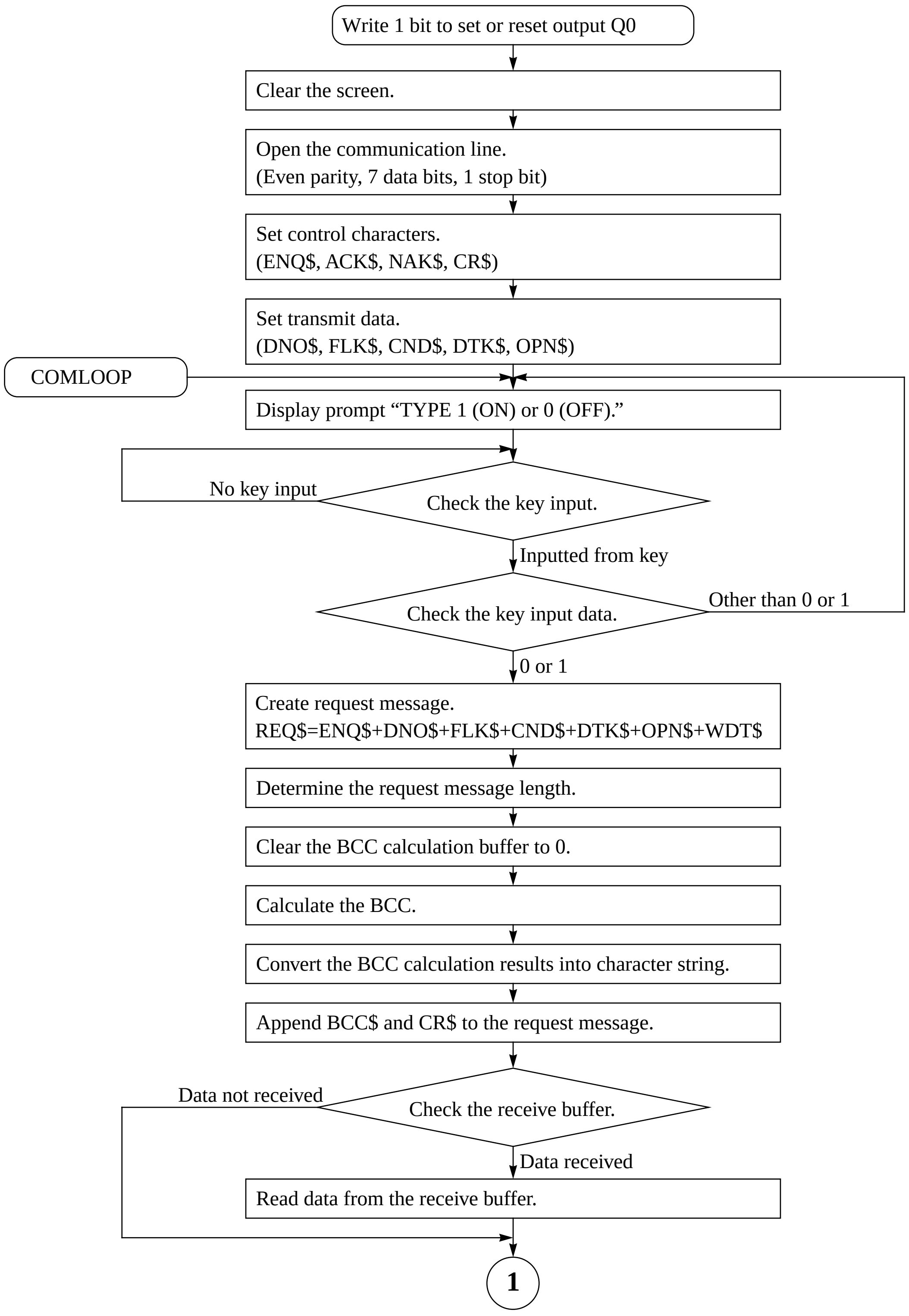
```
1000 'SAVE "RDTBYT.BAS",A
1010 '+---------------------------------------------+
1020 '|                                             |
1030 '|         Read Timer/Counter Current Value    |
1040 '|                                             |
1050 '|         Timer T0 Current Value              |
1060 '|                                             |
1070 '+---------------------------------------------+
1080 '
10000 CLS
10010 '---------------------------------------- [Open Communication Line]
10020 OPEN "COM1:9600,E,7,1" AS #1            'Even parity, 7 data bits, 1 stop
10030 '---------------------------------------- [Set Control Characters]
10040 ENQ$=CHR$(&H5)                          'Enquiry
10050 ACK$=CHR$(&H6)                          'Acknowledge
10060 NAK$=CHR$(&H15)                         'Negative acknowledge
10070 CR$ =CHR$(&HD)                          'Terminator code
10080 '---------------------------------------- [Set Transmit Data]
10090 DNO$="00"                               'Device number (00)
10100 FLK$="0"                                'Continuation (Discontinued)
10110 CND$="R"                                'Command (Read data)
10120 DTK$="t"                                'Data type (Timer current value)
10130 OPN$="0000"                             'Operand number (0000)
10140 DLN$="02"                               'Data length (2 bytes)
10150 '---------------------------------------- [Create Request Message]
10160 REQ$=ENQ$+DNO$+FLK$+CND$+DTK$+OPN$+DLN$   'BCC calculation range
10170 REQL=LEN(REQ$)                          'Determine BCC calculation range
10180 BCC=0                                   'Initialize BCC
10190 FOR I=1 TO REQL                         'Repeat BCC calculation range
10200   BCC=BCC XOR ASC(MID$(REQ$,I,1))       'Calculate BCC
10210 NEXT I
10220 BCC$=RIGHT$("0"+HEX$(BCC),2)            'Convert BCC results into character string
10230 REQ$=REQ$+BCC$+CR$                      'Append BCC and CR to request message
10240 '---------------------------------------- [Send Request Message]
10250 'COMLOOP
10260 IF LOC(1)<>0 THEN DUM$=INPUT$(LOC(1),#1)  'Clear receive buffer if not empty
10270 OUT &H3FC,( INP(&H3FC) OR 2 )           'Enable to send
10280 PRINT #1,REQ$;                          'Write into transmit buffer
10290 IF INP(&H3FD)=&H60 THEN 10300 ELSE 10290  'Wait until transmit buffer is empty
10300 OUT &H3FC,( INP(&H3FC) AND 253 )        'Prohibit to send
10310 '---------------------------------------- [Receive Reply Message]
10320 FOR WAITA=1 TO 500                      'Wait for receive data
10330   IF LOC(1)>=11 THEN GOTO 10390         'Check received data bytes
10340 NEXT WAITA
10350 IF LOC(1)<>0 THEN GOTO 10390            'Jump if receive buffer is not empty
10360 REP$=""                                 'Clear the reply message
10370 GOTO 10590                              'Display timeout error
10380 '---------------------------------------- [Check Reply Message]
10390 'RDCHK
10400 REPLY$=INPUT$(LOC(1),#1)                'Read from receive buffer
10410 IF LEFT$(REPLY$,1)=ACK$ THEN GOTO 10450   'Check the first character of received data
10420 IF LEFT$(REPLY$,1)=NAK$ THEN GOTO 10660
10430 GOTO 10750
10440 '---------------------------------------- [Display Current Value]
10450 'DATDSP
10460 IF MID$(REPLY$,4,1)<>"0" THEN GOTO 10820
10470 D1=VAL(MID$(REPLY$,5,1))                'Extract the timeout/countout bit
10480 TUP=D1 AND 8
10490 IF TUP=0 THEN TUP$="OFF" ELSE TUP$="ON "
10500 DAT=VAL("&h"+MID$(REPLY$,5,4))          'Convert binary to BCD
10510 DAT=DAT AND &H3FFF                      'Clear upper 2 bits to zero
10520 LOCATE 10,10
10530 PRINT CHR$(ASC(DTK$)-32);OPN$;"  ";     'Display current value
10540 PRINT USING "#####";DAT
10550 LOCATE 10,25
```

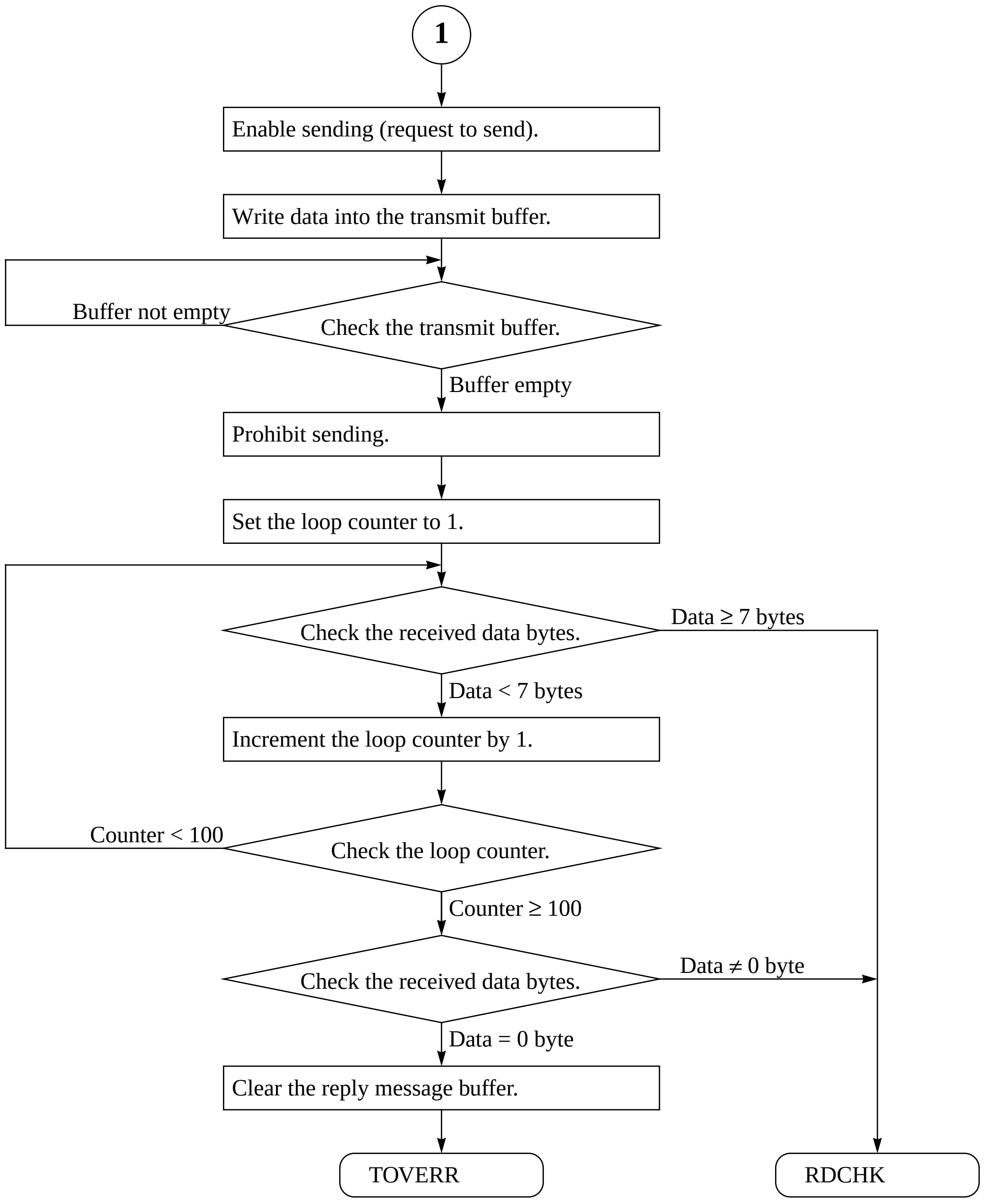
```
10560 PRINT "  STATUS ";TUP$                    'Display timeout/countout status
10570 GOTO 10250
10580 '---------------------------------------- [Display Timeout Error]
10590 'TOVERR
10600 CLS
10610 LOCATE 10,10
10620 PRINT "    Receive Timeout Error !    "
10625 BEEP
10630 FOR WAITB=1 TO 5000 : NEXT WAITB
10640 CLS : GOTO 10250
10650 '---------------------------------------- [Display NAK Receive Error]
10660 'NAKERR
10670 CLS :
10680 LOCATE 10,10
10690 PRINT "    NAK Receive Error !   "
10695 BEEP
10700 LOCATE 14,12
10710 PRINT "ERROR CODE=";MID$(REPLY$,5,2)
10720 FOR WAITC=1 TO 5000 : NEXT WAITC
10730 CLS : GOTO 10250
10740 '---------------------------------------- [Display Communication Error]
10750 'COMERR
10760 CLS : BEEP
10770 LOCATE 10,10
10780 PRINT "    Communication Error !    "
10790 FOR WAITD=1 TO 5000 : NEXT WAITD
10800 CLS : GOTO 10250
10810 '---------------------------------------- [Display NG Reply Error]
10820 'NGERR
10830 CLS : BEEP
10840 LOCATE 10,10
10850 PRINT "   NG Reply Error !    ""
10860 LOCATE 14,12
10870 PRINT "ERROR CODE=";MID$(REPLY$,5,2)
10880 FOR WAITE=1 TO 5000 : NEXT WAITE
10890 CLS : GOTO 10250
```

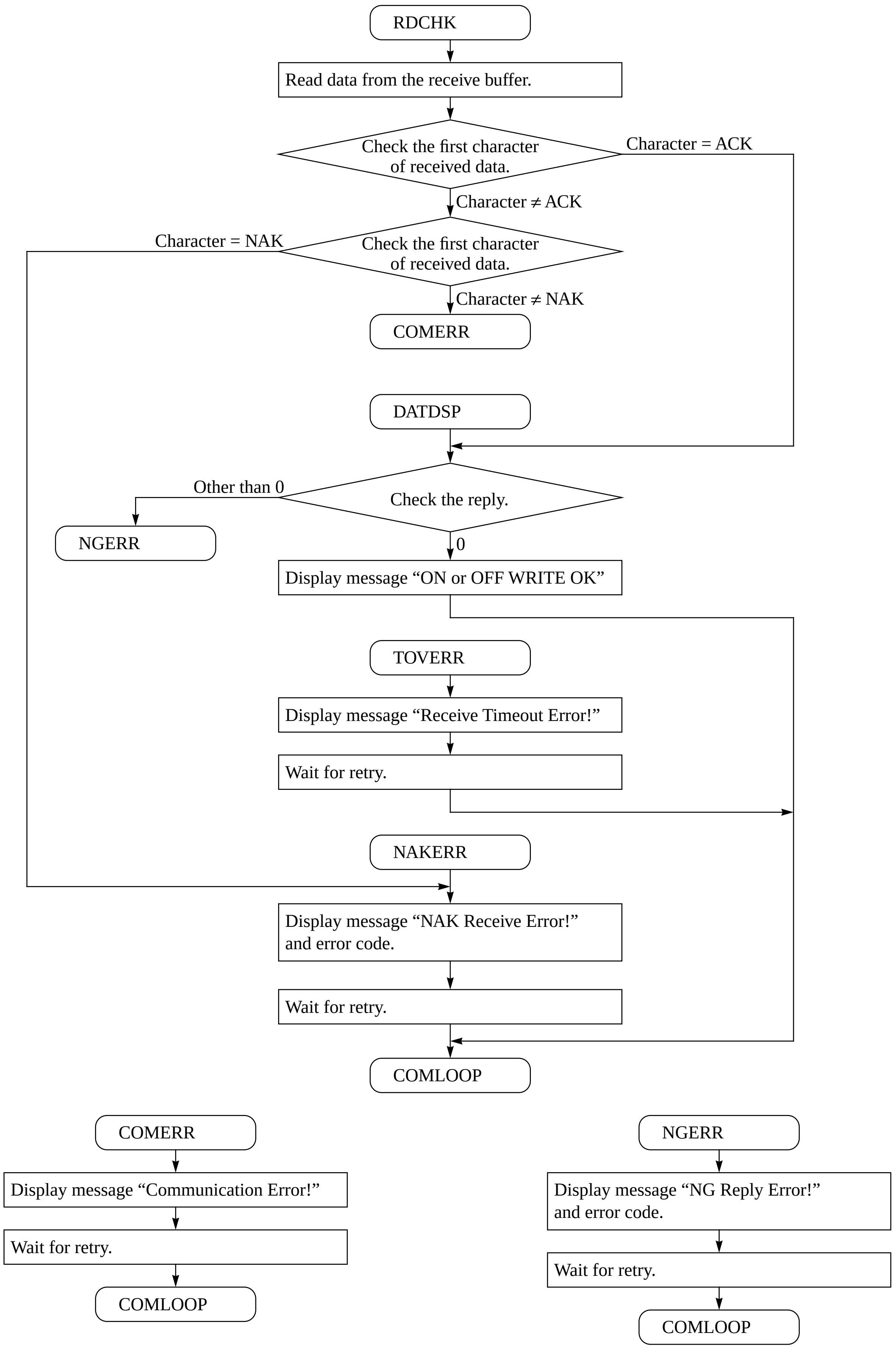
Write 1 Bit to Set or Reset Output Q0

This example demonstrates a program to turn output Q0 on or off of the MICRO³ of device number 0 when a 1 or 0 is entered from the computer keyboard, respectively. Setting or resetting the output is possible only while MICRO³ is running. If output Q0 is included in the user program, actual output Q0 is driven according to the user program at the END instruction although output Q0 is set or reset internally.

Flow Chart for Writing 1 Bit to Set or Reset Output Q0


Write 1 bit to set or reset output Q0
Clear the screen.
Open the communication line.
(Even parity, 7 data bits, 1 stop bit)
Set control characters.
(ENQ$, ACK$, NAK$, CR$)
Set transmit data.
(DNO$, FLK$, CND$, DTK$, OPN$)
COMLOOP
Display prompt "TYPE 1 (ON) or 0 (OFF)."
No key input
Check the key input.
Inputted from key
Check the key input data.
Other than 0 or 1
0 or 1
Create request message.
REQ$=ENQ$+DNO$+FLK$+CND$+DTK$+OPN$+WDT$
Determine the request message length.
Clear the BCC calculation buffer to 0.
Calculate the BCC.
Convert the BCC calculation results into character string.
Append BCC$ and CR$ to the request message.
Data not received
Check the receive buffer.
Data received
Read data from the receive buffer.
1


1
Enable sending (request to send).
Write data into the transmit buffer.
Buffer not empty
Check the transmit buffer.
Buffer empty
Prohibit sending.
Set the loop counter to 1.
Check the received data bytes.
Data ≥ 7 bytes
Data < 7 bytes
Increment the loop counter by 1.
Counter < 100
Check the loop counter.
Counter ≥ 100
Check the received data bytes.
Data ≠ 0 byte
Data = 0 byte
Clear the reply message buffer.
TOVERR
RDCHK


RDCHK
Read data from the receive buffer.
Check the first character of received data.
Character = ACK
Character ≠ ACK
Check the first character of received data.
Character = NAK
Character ≠ NAK
COMERR
DATDSP
Check the reply.
Other than 0
NGERR
0
Display message "ON or OFF WRITE OK"
TOVERR
Display message "Receive Timeout Error!"
Wait for retry.
NAKERR
Display message "NAK Receive Error!" and error code.
Wait for retry.
COMLOOP
COMERR
Display message "Communication Error!"
Wait for retry.
COMLOOP
NGERR
Display message "NG Reply Error!" and error code.
Wait for retry.
COMLOOP

Program List (Filename: WT1BIT.BAS)

When executing this sample program, set the MICRO^3 communication format to the default values and the computer communication baud rate to 9600 bps.

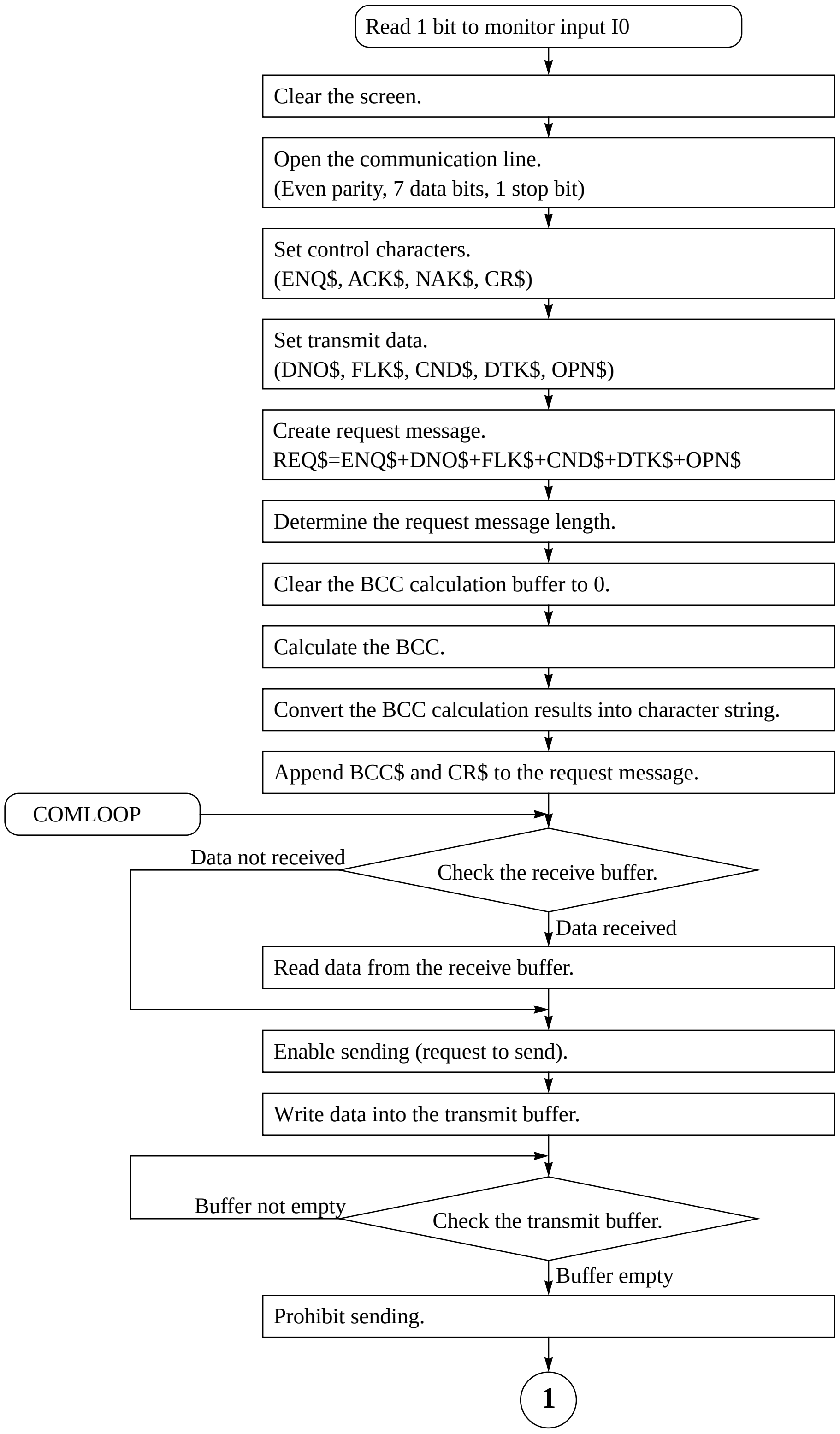
```
1000 'SAVE "WT1BIT.BAS",A
1010 '+--------------------------------------------+
1020 '|                                            |
1030 '|         Write 1 Bit                        |
1040 '|                                            |
1050 '|         Write to Output Q0                 |
1060 '|                                            |
1070 '+--------------------------------------------+
1080 '
10000 CLS
10010 '----------------------------------------- [Open Communication Line]
10020 OPEN "COM1:9600,E,7,1" AS #1            'Even parity, 7 data bits, 1 stop bit
10030 '----------------------------------------- [Set Control Characters]
10040 ENQ$=CHR$(&H5)                          'Enquiry
10050 ACK$=CHR$(&H6)                          'Acknowledge
10060 NAK$=CHR$(&H15)                         'Negative acknowledge
10070 CR$ =CHR$(&HD)                          'Terminator code
10080 '----------------------------------------- [Set Transmit Data]
10090 DNO$="00"                               'Device number (00)
10100 FLK$="0"                                'Continuation (Discontinued)
10110 CND$="W"                                'Command (Write data)
10120 DTK$="y"                                'Data type (Output)
10130 OPN$="0000"                             'Operand number (0000)
10140 '----------------------------------------- [Create Request Message]
10150 'COMLOOP
10160 LOCATE 10,6
10170 PRINT " TYPE  1(ON) or 0(OFF) : ";
10180 WDT$=INKEY$:IF WDT$="" THEN GOTO 10180    'Wait for key input
10190 PRINT WDT$                              'Display ON/OFF setting
10200 IF WDT$="0" OR WDT$="1" THEN GOTO 10230   'Check input data
10210 BEEP:LOCATE 10,31:PRINT " "
10220 GOTO 10150                              'Retry key input
10230 'MKCOM
10240 REQ$=ENQ$+DNO$+FLK$+CND$+DTK$+OPN$+WDT$   'BCC calculation range
10250 REQL=LEN(REQ$)                          'Determine BCC calculation range
10260 BCC=0                                   'Initialize BCC
10270 FOR I=1 TO REQL                         'Repeat BCC calculation range
10280   BCC=BCC XOR ASC(MID$(REQ$,I,1))       'Calculate BCC
10290 NEXT I
10300 BCC$=RIGHT$("0"+HEX$(BCC),2)            'Convert BCC results into character string
10310 REQ$=REQ$+BCC$+CR$                      'Append BCC and CR to request message
10320 '----------------------------------------- [Send Request Message]
10330 IF LOC(1)<>0 THEN DUM$=INPUT$(LOC(1),#1)  'Clear receive buffer if not empty
10340 OUT &H3FC,( INP(&H3FC) OR 2 )           'Enable to send
10350 PRINT #1,REQ$;                          'Write into transmit buffer
10360 IF INP(&H3FD)=&H60 THEN 10370 ELSE 10360  'Wait until transmit buffer is empty
10370 OUT &H3FC,( INP(&H3FC) AND 253 )        'Prohibit to send
10380 '----------------------------------------- [Receive Reply Message]
10390 FOR WAITA=1 TO 500                      'Wait for receive data
10400   IF LOC(1)>=7 THEN GOTO 10460          'Check received data bytes
10410 NEXT WAITA
10420 IF LOC(1)<>0 THEN GOTO 10460            'Jump if receive buffer is not empty
10430 REP$=""                                 'Clear the reply message
10440 GOTO 10600                              'Display timeout error
10450 '----------------------------------------- [Check Reply Message]
10460 'RDCHK
10470 REPLY$=INPUT$(LOC(1),#1)                'Read from receive buffer
10480 IF LEFT$(REPLY$,1)=ACK$ THEN GOTO 10520   'Check the first character of received data
10490 IF LEFT$(REPLY$,1)=NAK$ THEN GOTO 10670
10500 GOTO 10760
10510 '----------------------------------------- [Display ON/OFF Results]
10520 'DATDSP
10530 IF MID$(REPLY$,4,1)<>"0" THEN GOTO 10830  'Check ON/OFF results
10540 LOCATE 11,10
10550 PRINT "Q";OPN$;"  ";
```

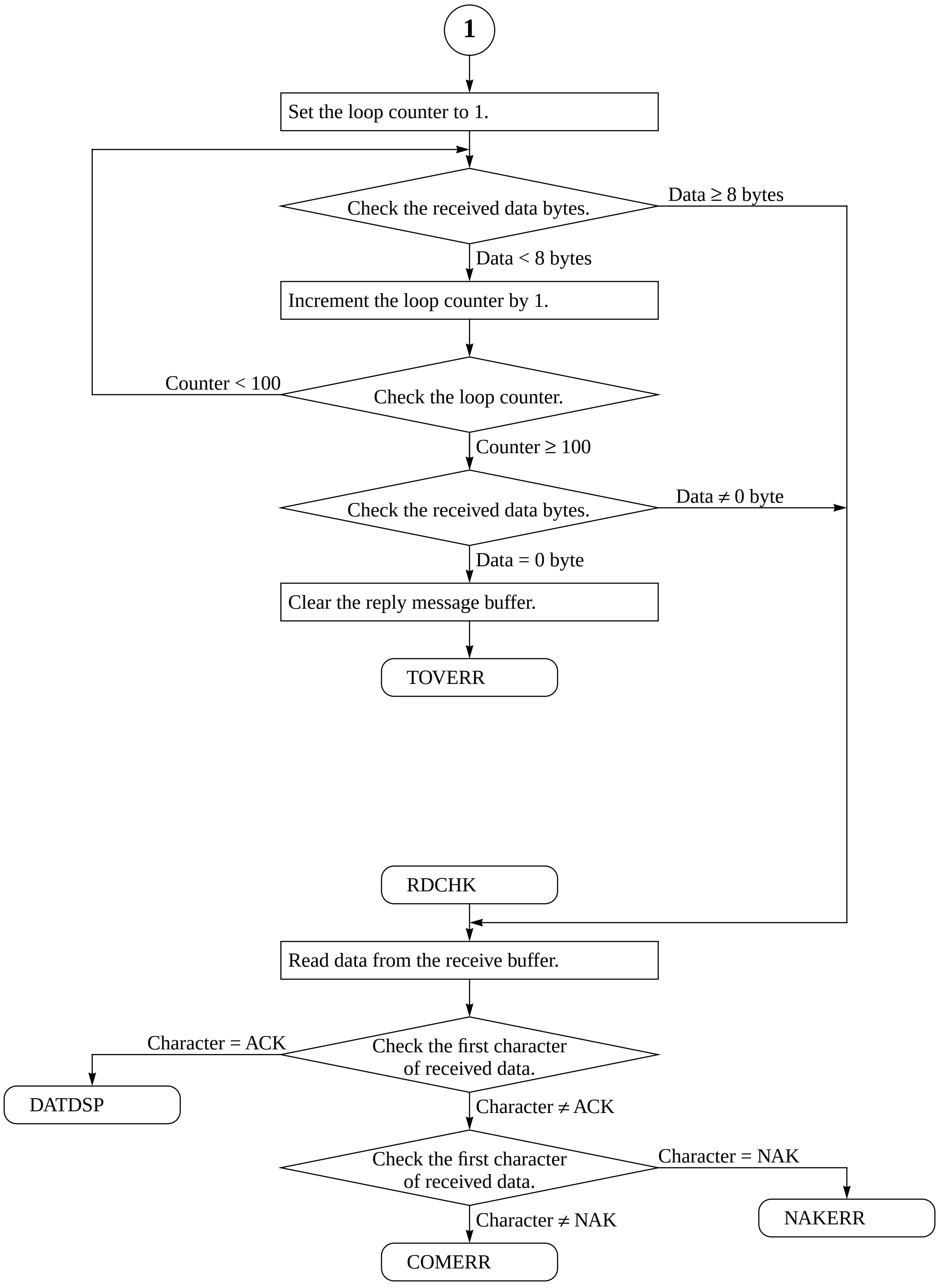
```
10560 IF WDT$="0" THEN ST$="OFF" ELSE ST$="ON "
10570 PRINT ST$; "    - WRITE OK -"
10580 GOTO 10150
10590 '----------------------------------------- [Display Timeout Error]
10600 'TOVERR
10610 CLS : BEEP
10620 LOCATE 10,10
10630 PRINT "    Receive Timeout Error !    "
10640 FOR WAITB=1 TO 5000 : NEXT WAITB
10650 CLS : GOTO 10150
10660 '----------------------------------------- [Display NAK Receive Error]
10670 'NAKERR
10680 CLS : BEEP
10690 LOCATE 10,10
10700 PRINT "    NAK Receive Error !   "
10710 LOCATE 14,12
10720 PRINT "ERROR CODE=";MID$(REPLY$,5,2)
10730 FOR WAITC=1 TO 5000 : NEXT WAITC
10740 CLS : GOTO 10150
10750 '----------------------------------------- [Display Communication Error]
10760 'COMERR
10770 CLS : BEEP
10780 LOCATE 10,10
10790 PRINT "    Communication Error !    "
10793 LOCATE 11,10
10795 PRINT "Reply=";REPLY$
10800 FOR WAITD=1 TO 15000 : NEXT WAITD
10810 CLS : GOTO 10150
10820 '----------------------------------------- [Display NG Reply Error]
10830 'NGERR
10840 CLS : BEEP
10850 LOCATE 10,10
10860 PRINT "   NG Reply Error !    "
10870 LOCATE 14,12
10880 PRINT "ERROR CODE=";MID$(REPLY$,5,2)
10890 FOR WAITE=1 TO 5000 : NEXT WAITE
10900 CLS : GOTO 10150
```

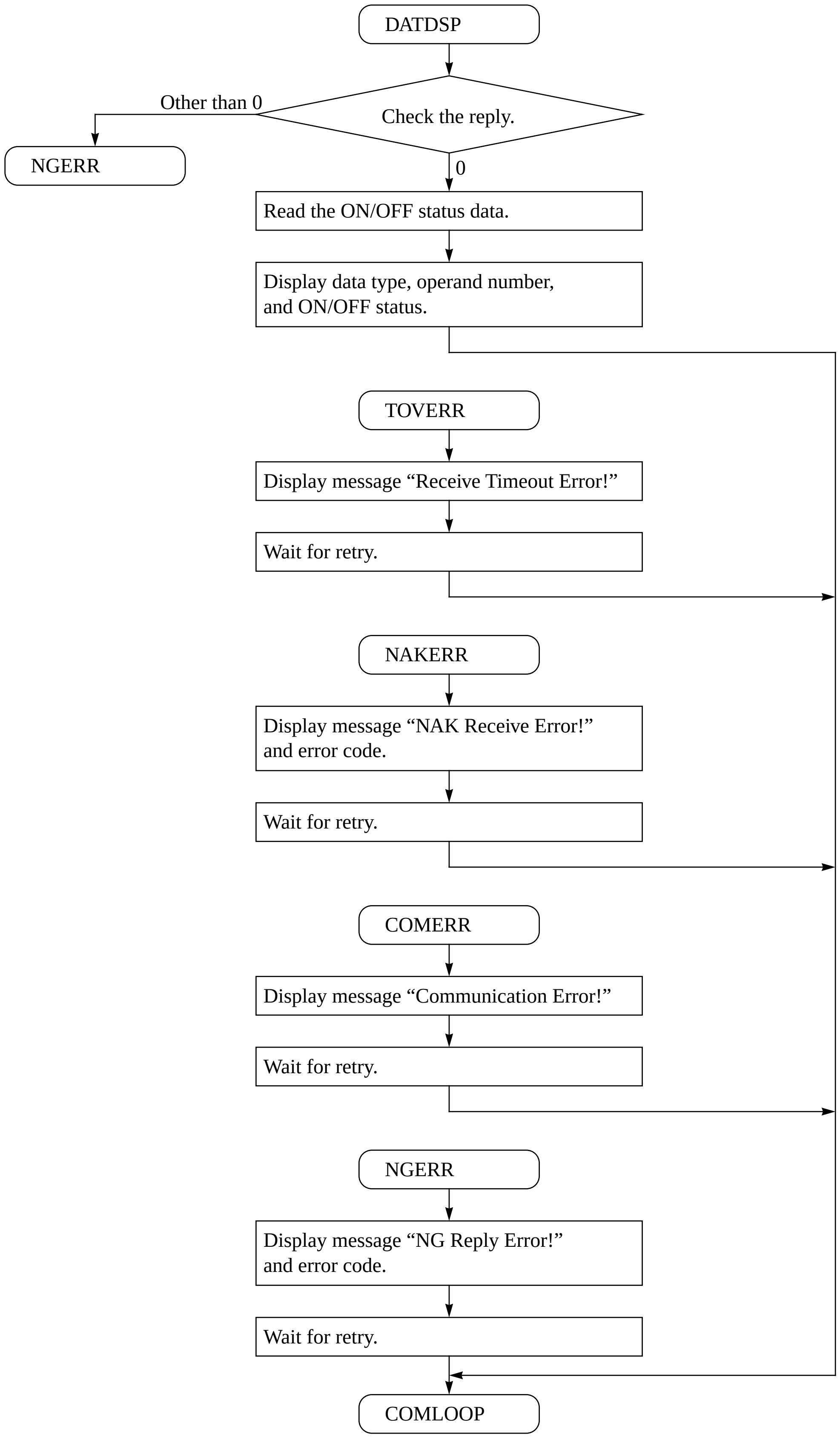
Read 1 Bit to Monitor Input I0

This example demonstrates a program to monitor the status of input I0 in the MICRO3 of device number 0 and display the monitored ON/OFF status on the computer screen. Monitoring the input status is possible whether MICRO3 is running or stopped.

Flow Chart for Reading 1 Bit to Monitor Input I0


Read 1 bit to monitor input I0
Clear the screen.
Open the communication line.
(Even parity, 7 data bits, 1 stop bit)
Set control characters.
(ENQ$, ACK$, NAK$, CR$)
Set transmit data.
(DNO$, FLK$, CND$, DTK$, OPN$)
Create request message.
REQ$=ENQ$+DNO$+FLK$+CND$+DTK$+OPN$
Determine the request message length.
Clear the BCC calculation buffer to 0.
Calculate the BCC.
Convert the BCC calculation results into character string.
Append BCC$ and CR$ to the request message.
COMLOOP
Data not received
Check the receive buffer.
Data received
Read data from the receive buffer.
Enable sending (request to send).
Write data into the transmit buffer.
Buffer not empty
Check the transmit buffer.
Buffer empty
Prohibit sending.
1


1
Set the loop counter to 1.
Check the received data bytes.
Data ≥ 8 bytes
Data < 8 bytes
Increment the loop counter by 1.
Counter < 100
Check the loop counter.
Counter ≥ 100
Check the received data bytes.
Data ≠ 0 byte
Data = 0 byte
Clear the reply message buffer.
TOVERR
RDCHK
Read data from the receive buffer.
Character = ACK
Check the first character of received data.
DATDSP
Character ≠ ACK
Check the first character of received data.
Character = NAK
NAKERR
Character ≠ NAK
COMERR


DATDSP
Check the reply.
Other than 0
NGERR
0
Read the ON/OFF status data.
Display data type, operand number, and ON/OFF status.
TOVERR
Display message "Receive Timeout Error!"
Wait for retry.
NAKERR
Display message "NAK Receive Error!" and error code.
Wait for retry.
COMERR
Display message "Communication Error!"
Wait for retry.
NGERR
Display message "NG Reply Error!" and error code.
Wait for retry.
COMLOOP

Program List (Filename: RD1BIT.BAS)

When executing this sample program, set the MICRO^3 communication format to the default values and the computer communication baud rate to 9600 bps.

```
1000 'SAVE "RD1BIT.BAS",A
1010 '+------------------------------------------+
1020 '|                                          |
1030 '|         Read 1 Bit                       |
1040 '|                                          |
1050 '|         Read from Input I0               |
1060 '|                                          |
1070 '+------------------------------------------+
1080 '
10000 CLS
10010 '------------------------------------------ [Open Communication Line]
10020 OPEN "COM1:9600,E,7,1" AS #1             'Even parity, 7 data bits, 1 stop bit
10030 '------------------------------------------ [Set Control Characters]
10040 ENQ$=CHR$(&H5)                           'Enquiry
10050 ACK$=CHR$(&H6)                           'Acknowledge
10060 NAK$=CHR$(&H15)                          'Negative acknowledge
10070 CR$ =CHR$(&HD)                           'Terminator code
10080 '------------------------------------------ [Set Transmit Data]
10090 DNO$="00"                                'Device number (00)
10100 FLK$="0"                                 'Continuation (Discontinued)
10110 CND$="R"                                 'Command (Read data)
10120 DTK$="x"                                 'Data type (Input)
10130 OPN$="0000"                              'Operand number (0000)
10140 '------------------------------------------ [Create Request Message]
10150 REQ$=ENQ$+DNO$+FLK$+CND$+DTK$+OPN$        'BCC calculation range
10160 REQL=LEN(REQ$)                           'Determine BCC calculation range
10170 BCC=0                                    'Initialize BCC
10180 FOR I=1 TO REQL                          'Repeat BCC calculation range
10190   BCC=BCC XOR ASC(MID$(REQ$,I,1))        'Calculate BCC
10200 NEXT I
10210 BCC$=RIGHT$("0"+HEX$(BCC),2)             'Convert BCC results into character string
10220 REQ$=REQ$+BCC$+CR$                       'Append BCC and CR to request message
10230 '------------------------------------------ [Send Request Message]
10240 'COMLOOP
10250 IF LOC(1)<>0 THEN DUM$=INPUT$(LOC(1),#1)  'Clear receive buffer if not empty
10260 OUT &H3FC,( INP(&H3FC) OR 2 )            'Enable to send
10270 PRINT #1,REQ$;                           'Write into transmit buffer
10280 IF INP(&H3FD)=&H60 THEN 10290 ELSE  10280 'Wait until transmit buffer is empty
10290 OUT &H3FC,( INP(&H3FC) AND 253 )         'Prohibit to send
10300 '------------------------------------------ [Receive Reply Message]
10310 FOR WAITA=1 TO 500                       'Wait for receive data
10320   IF LOC(1)>=8 THEN GOTO 10380           'Check received data bytes
10330 NEXT WAITA
10340 IF LOC(1)<>0 THEN GOTO 10380             'Jump if receive buffer is not empty
10350 REP$=""                                  'Clear the reply message
10360 GOTO 10530                               'Display timeout error
10370 '------------------------------------------ [Check Reply Message]
10380 'RDCHK
10390 REPLY$=INPUT$(LOC(1),#1)                 'Read from receive buffer
10400 IF LEFT$(REPLY$,1)=ACK$ THEN GOTO 10440   'Check the first character of received data
10410 IF LEFT$(REPLY$,1)=NAK$ THEN GOTO 10600
10420 GOTO 10700
10430 '------------------------------------------ [Display ON/OFF Status]
10440 'DATDSP
10450 IF MID$(REPLY$,4,1)<>"0" THEN GOTO 10760
10460 DAT$=MID$(REPLY$,5,1)                    'Extract ON/OFF status
10470 LOCATE 10,10
10480 PRINT "I";OPN$;"  ";
10490 IF DAT$="0" THEN ST$="OFF" ELSE ST$="ON "
10500 PRINT ST$
10510 GOTO 10240
10520 '------------------------------------------ [Display Timeout Error]
10530 'TOVERR
10540 CLS
10550 LOCATE 10,10
```

```
10560 PRINT "    Receive Timeout Error !   "
10565 BEEP
10570 FOR WAITB=1 TO 5000 : NEXT WAITB
10580 CLS : GOTO 10240
10590 '--------------------------------------- [Display NAK Receive Error]
10600 'NAKERR
10610 CLS : BEEP
10620 LOCATE 10,10
10630 PRINT "    NAK Receive Error !   "
10640 LOCATE 14,12
10650 PRINT "ERROR CODE=";MID$(REPLY$,5,2)
10660 'FOR WAITC=1 TO 5000 : NEXT WAITC
10665 TMP$=INPUT$(1)
10670 CLS : GOTO 10240
10680 '--------------------------------------- [Display Communication Error]
10690 'COMERR
10700 CLS : BEEP
10710 LOCATE 10,10
10720 PRINT "    Communication Error !    "
10730 FOR WAITD=1 TO 5000 : NEXT WAITD
10740 CLS : GOTO 10240
10750 '--------------------------------------- [Display NG Reply Error]
10760 'NGERR
10770 CLS : BEEP
10780 LOCATE 10,10
10790 PRINT "   NG Reply Error !    "
10800 LOCATE 14,12
10810 PRINT "ERROR CODE=";MID$(REPLY$,5,2)
10820 FOR WAITE=1 TO 5000 : NEXT WAITE
10830 CLS : GOTO 10240
```

Communication Troubles

This chapter describes probable troubles during communication between the computer and MICRO^3 and explains how to correct the troubles. Troubles 1 through 3 relate to communication failures and trouble 4 is an NG reply message returned during normal communication.

Trouble 1

After sending a request message, no reply message is returned.

Probable Cause and Action

- The communication line such as the twisted pair cable is disconnected.

  Make sure that all cables are connected correctly.

- When sending the request message, the interval between character codes exceeded the receive timeout value of the MICRO^3. The default receive timeout value is 500 msec.

  Make sure that the receive timeout value is not exceeded between characters when sending the request message. Or, extend the receive timeout value for MICRO^3 using FUN8. See page 3-3.

- The programmed device number in the request message is not found in any MICRO^3 connected in the communication network. When the programmed device number is not found in the communication network, no reply message is returned from any MICRO^3 unit.

  Check the device numbers in all MICRO^3 units connected in the network using FUN9. Make sure that the device number settings match between the request message and MICRO^3.

- The communication format does not match between the computer and MICRO^3.

  Make sure of the same communication format for the computer and MICRO^3. When the communication format for MICRO^3 has been changed using FUN8 (loader port communication mode setting), turn the programmed mode selection input on to enable the modified communication format.

Trouble 2

The received reply message does not meet the reply message format.

Probable Cause and Action

- The communication data was changed by noise during communication.

  Enhance the cable shielding and grounding to make sure that the cable is not influenced by noise.

- The communication format does not match between the computer and MICRO^3.

  Make sure of the same communication format for the computer and MICRO^3. When the communication format for MICRO^3 has been changed using FUN8 (loader port communication mode setting), turn the programmed mode selection input on to enable the modified communication format.

Trouble 3

A NAK (15h) reply message is returned, which signals that an error occurred during communication of the request message. See page 3-5.

Probable Cause and Action

- The communication data was changed by noise during communication.

  Enhance the cable shielding and grounding to make sure that the cable is not influenced by noise.

- The request message format is incorrect.

  Make sure of the correct request message. See chapters 3 and 4.

- When error code 00 is included in the reply message, the BCC code calculation is incorrect.

  Make sure of correct BCC code. See chapter 5.

Trouble 4

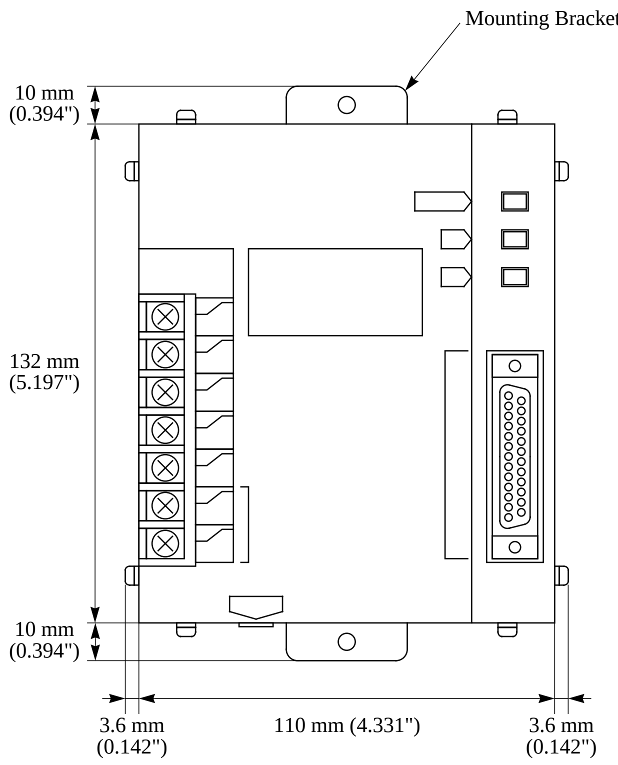
An NG reply is returned. The first character of the reply message is ACK (06h) and the command code is 2 (32h) which means NG (error).

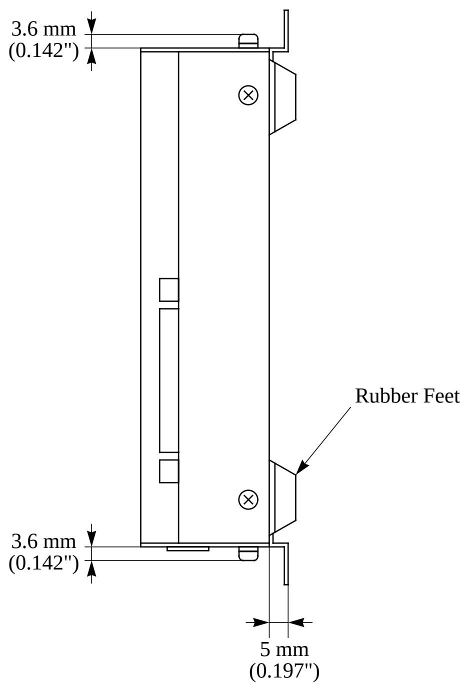
Probable Cause and Action

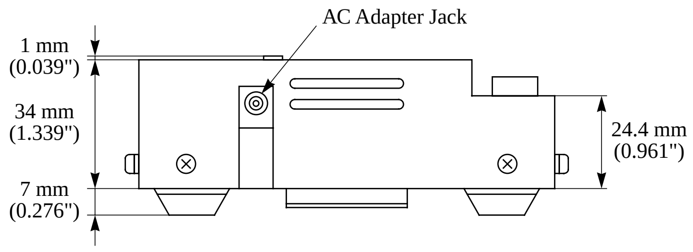
The reply message signals an error. Check the NG code and take a corrective action shown in the table below:

NG Code	Cause	Action
00 (Expansion station error)	Communication was attempted to the expansion station.	Communicate to the base station. Set the function selector switch on MICRO^3 to 0.
01 (Program size error)	Improper write/read program size. When writing a user program to MICRO^3, the user program capacity is larger than the program capacity selected in MICRO^3. When reading a user program from MICRO^3, the program capacity selected in MICRO^3 is larger than the user program receive buffer in the computer.	Check the program capacity setting in MICRO^3 using FUN11. Send a user program smaller than or equal to the program capacity setting. Increase the user program receive buffer capacity and send to MICRO^3 a request message including the capacity data.
02 (Protect error)	The user program in MICRO^3 is read and/or write protected.	Cancel the program protection using FUN22 on the program loader or CUBIQ.
03 (RUN error)	Writing user program attempted while MICRO^3 is running.	Stop MICRO^3 and try writing user program to MICRO^3 again.
04 (CRC error)	User program CRC code not matched. The user program to be written is broken.	Correct the user program and send the corrected user program to MICRO^3.
05 (Protect code error)	Protect code in the request message does not match that set in FUN27. Attempt was made to enable protection on a protected user program.	Send a correct protect code to the PLC. Do not attempt to enable protection on a protected user program.
06 (Data range error)	Designated data range is invalid.	Make sure of the correct data range. See MICRO^3 user's manual EM289.
07 (Timer/counter preset value change error)	Preset value change attempted to timer or counter with preset value designated by data register.	Timer/counter preset values in MICRO^3 can not be changed when a data register is designated as a preset value. Check the user program in MICRO^3 to see that the timer/counter has a constant designated as a preset value. To change a timer or counter preset value designated by a data register, change the value of the data register.
08 (Calendar/clock data error)	Writing invalid value to calendar/clock attempted. The calendar/clock in MICRO^3 is broken.	Make sure of correct values for the calendar/clock.
09 (Data clear error)	Designated data cannot be cleared.	Check if an error has occurred in MICRO^3. Correct the error and try again.
10 (Data error)	Invalid data other than 0 (30h) - 9 (39h) or A (41h) - F (46h) is included in the request message.	Check the request message and send a correct request message.
11 (Setting error)	Incorrect setting for user communication (MICRO3C only)	Check the request message and send a correct request message.

RS232C / RS485 Converter FC2A-MD1

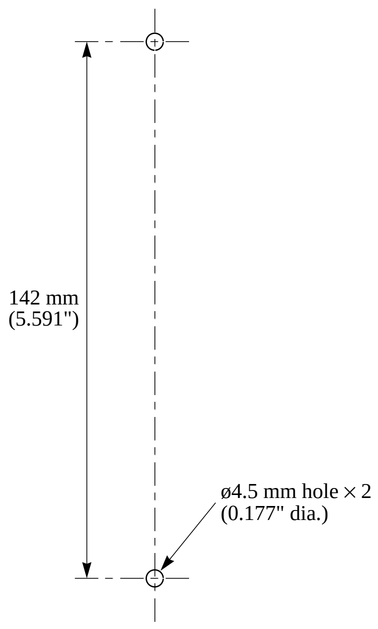




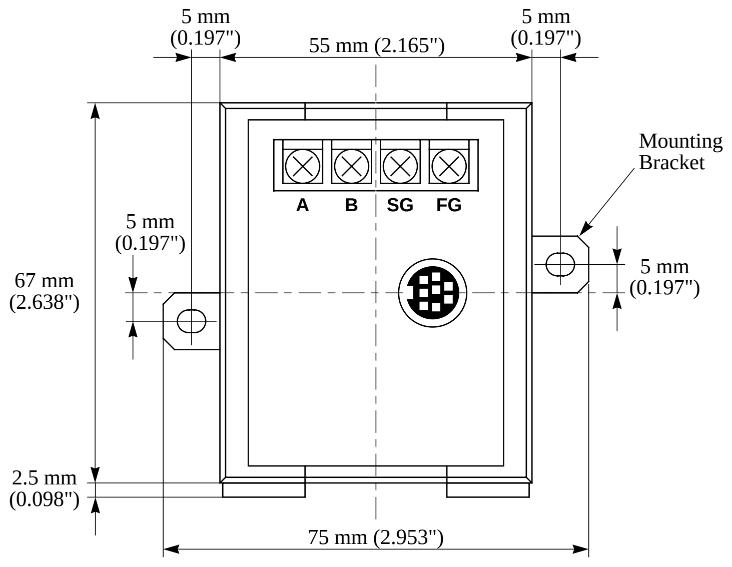


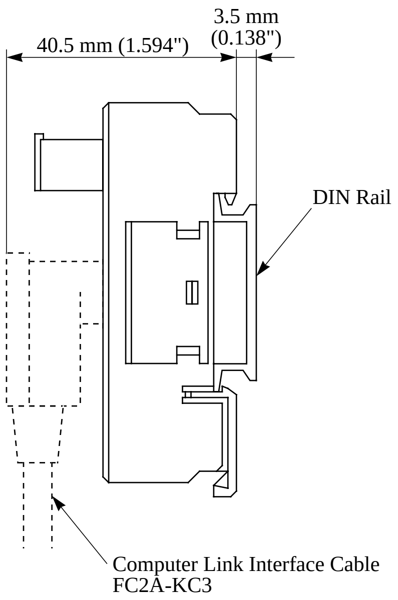
Note: When mounting the RS232C/RS485 converter on a panel surface, remove the rubber feet and attach the supplied mounting brackets on the bottom of the converter using screws.

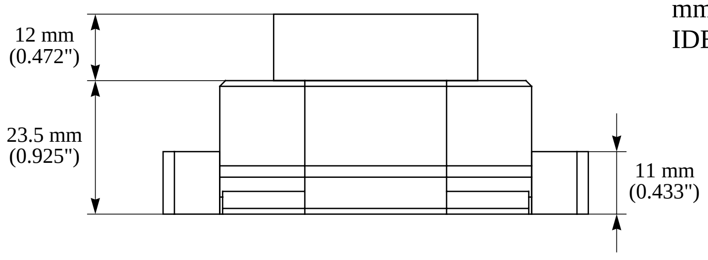
Mounting Hole Layout



Computer Link Interface Unit FC2A-LC1

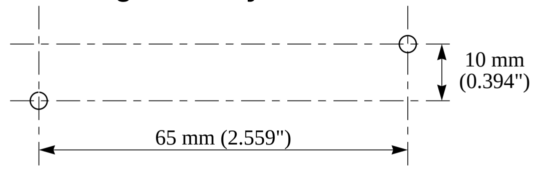




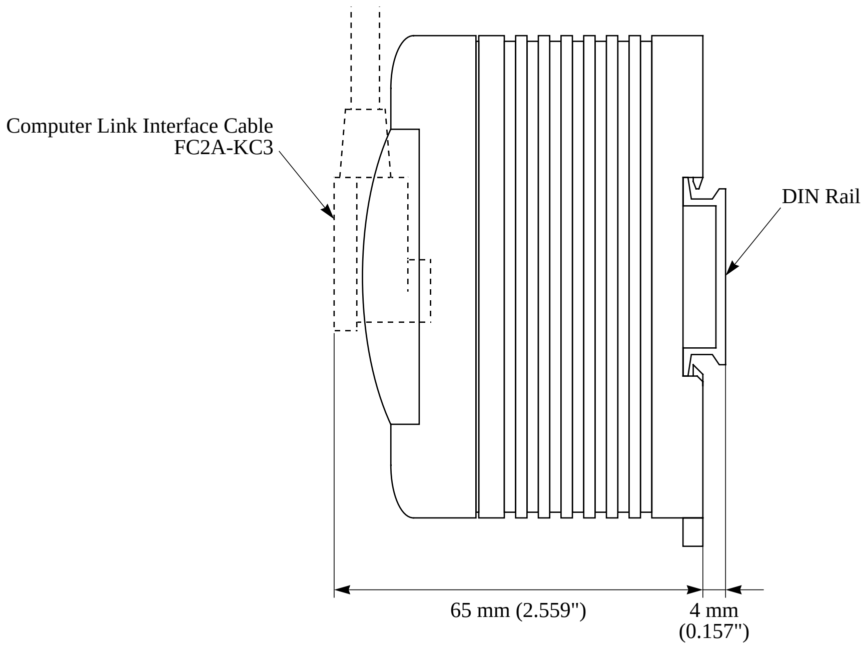


Note: To mount the computer link interface unit on a 35-mm-wide DIN rail, remove the mounting brackets. Use IDEC's BAA1000 DIN rail, 1000 mm (39.37") long.

Mounting Hole Layout



MICRO^3 Height with Computer Link Interface Cable


Computer Link Interface Cable FC2A-KC3
DIN Rail
65 mm (2.559")
4 mm (0.157")