AGB Dedicated IR Communication Adapter

# AGB Dedicated IR Communication Library Manual

Ver 1.0

Nintendo Co., Ltd.
Nintendo of America Inc.

## Revision History

| Version | Revision Date | Description of Revisions |
|---|---|---|
| 0.5 | 8/10/2000 | First Edition |
| 1.0 | 12/01/2000 | - Addition of irMountArmCore and irMountThumbCore. Also changed corresponding diagram.<br>- Combined library into libagbir.a.<br>- Changed the type of arguments for irRamCopy and irMountArmCore. (Changed defined values for IR_ARM_SEND_CORE_SIZE and IR_ARM_RECV_CORE_SIZE from byte units to words.)<br>- Changed so that the select timer number is with the irBegin function. Also changed the communication speed.<br>- Changed description in the “Cautions” section.<br>- Changed the title of this manual from “AGB Dedicated Protocol Specifications” to “AGB Dedicated IR Communication Library Manual”.<br>- Created the Table of Contents. |

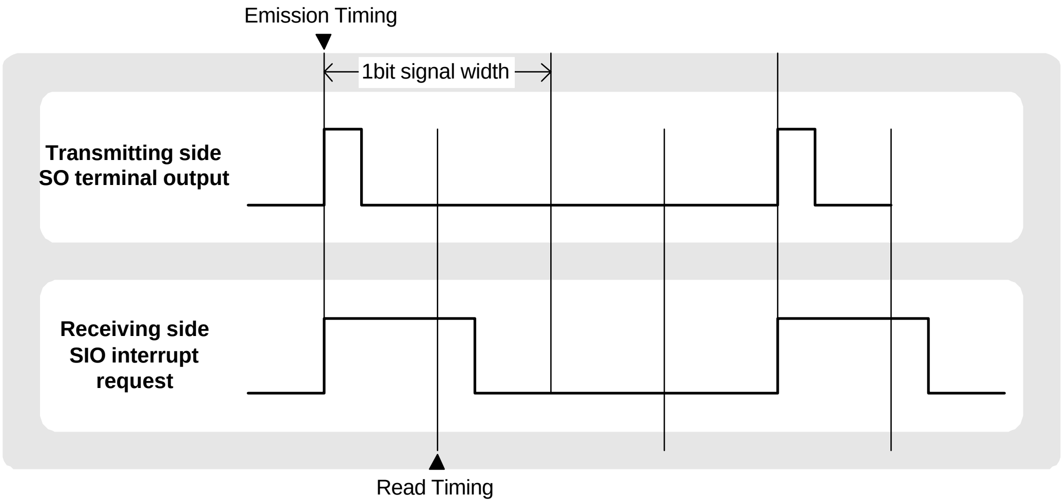
## Table of Contents

# 1.0 Specifications

## 1.1 Communication Protocol

The AGB dedicated protocol utilizes an RZ modulation method. Time is divided into fixed intervals, and the presence or absence of optical input during those intervals distinguishes signals.

A general schematic is shown below:



The 1bit signal width changes depending on the data communication speed.
*1bit signal width (μsec) = 1000 ÷ data transfer rate (kbps).*

## 1.2 Communication Speed

The data communication speed varies widely, depending on the ROM /RAM operating versions. Additionally, when using this version operation in ROM, the communication speed varies depending on the ROM access wait cycles.

| | |
|---|---|
| ROM version: | Approximately (or up to) ~ 47kbps |
| CPU internal RAM version: | Approximately 104.2kbps |

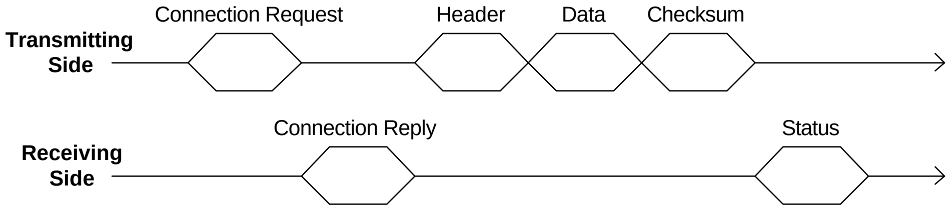
| ROM Wait Cycles | Communication speed [Arguments of irSendConnect, irRecvConnect] |
|---|---|
| 4 - 2 | 40kbps IR_40kbps_RATE |
| 3 - 1 | 47kbps IR_47kbps_RATE |
| Other | Not defined (Contact **support@noa.com** for support if needed.) |

**Version operating in CPU internal RAM**

| Communication speed [Arguments of irSendConnect, irRecvConnect] |
|---|
| 104.2kbps [IR_104kbps_RATE] |

## 2.0 Communication Procedure

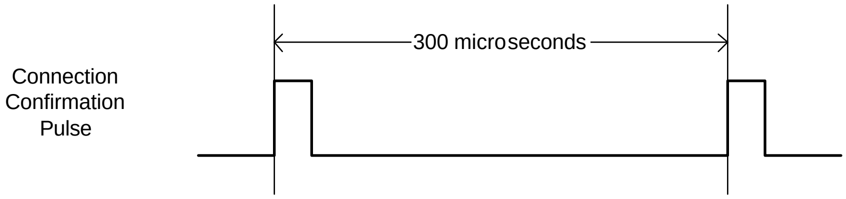
Communication proceeds as diagrammed below:



### 2.1 Connection Request/Connection Reply

Before the actual data transfer begins, a connection confirmation pulse is sent back and forth to confirm that the IR communication module is operating normally.

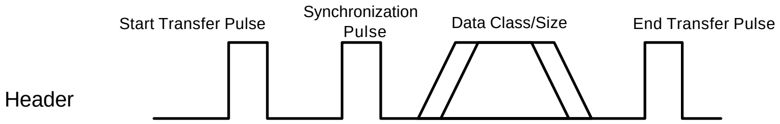
The receiving side receives the connection part of the signal. If it recognizes the connection pulse, it returns the same pulse back to the transmitting side. After that, the transmitting side determines whether the returned signal is a normal connection pulse. If the signal is not recognized to be a normal connection pulse, then communication will terminate.



### 2.2 Header

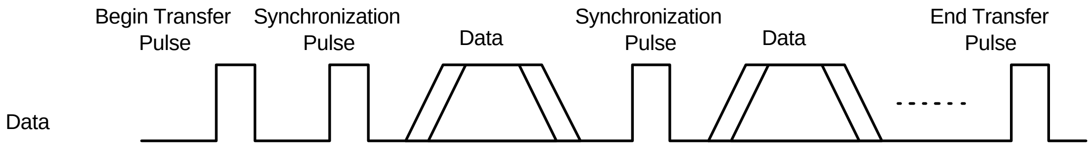
The header transmits the 1-halfword (16 bit) sum total of the data class (8bit) + data transfer size (8bit).

Communication will terminate if the data class is an unregistered illegal constant or if the data transfer size exceeds the permissible byte size of the receiving side.



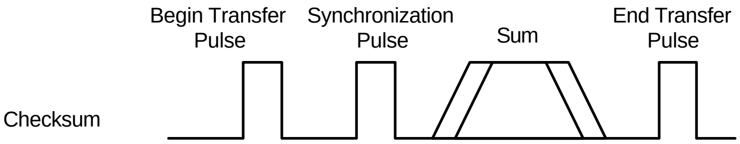
## 2.3 Data

The amount of data specified by the 8bit "data transfer size" part of the header is transmitted consecutively. The amount of data that can be transmitted consecutively is 0 ~ 255 halfwords.



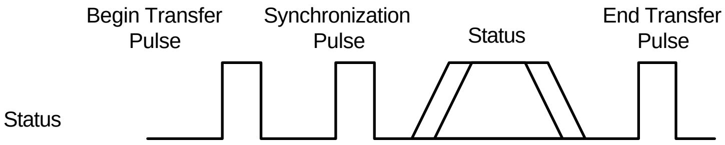
## 2.4 Checksum

This part transmits the checksum (1 halfword) calculated by the transmitting side.



## 2.5 Status

The receiving side returns its communication status (1 halfword).



### Table of Communication Status Values

| Status value | State | Explanation |
|---|---|---|
| IR_NORMAL | Normal | No error occurred |
| IR_PULSE_ERR | Pulse error | An abnormal pulse was received |
| IR_SUM_ERR | Checksum error | The checksum value sent by the transmitting side was not equal to the checksum value calculated by the receiving side. |
| IR_BYTES_ERR | Byte error | The transfer request data byte size exceeded the byte size permitted by the receiving side. |
| IR_CODE_ERR | Code error | The data class is an unregistered illegal constant. |

# 3.0 Programming

## 3.1 Function Reference

libagbir.a is the IR communication library. Make sure you include the 'AgbIr.h' header file. The IR communication library, libagbir.a, has the functions required for IR communication. The processing for each function is as follows:

void **irBegin**( u16 tmrno, irResult* IrResultStr, u8 wait)

Call this function at the beginning of IR communication. It internally backs up the IE, IME and timer control register for the timer "tmrno" in addition to the content of each register for the setup values. (By calling the function, irEnd, the back-up content is restored. The Interrupt Master Enable flag register (IME) is disabled to prohibit all interrupts. (If not, communication cannot be performed.)

In order to synchronize communication, the timer for the argument "tmrno" is set up to run. If a 1 is passed to the wait flag, **wait**, there is a delay of at least 2ms before communication is possible.

void **irRamCopy** ( u32* Ram_S_Buf, u32* Ram_R_Buf)

The core communication program for transmitting and receiving data (ARM code) is transferred to the buffer.

Make sure you set aside the following for the buffer in the **CPU internal RAM** which is 32 bit aligned:

- Required word size for send buffer (Ram_S_Buf): IR_ARM_SEND_CORE_SIZE
- Required word size for receive buffer (Ram_R_Buf): IR_ARM_RECV_CORE_SIZE

void **irMountArmCore**( u32* Ram_S_Buf, u32* Ram_R_Buf)

The send/receive buffer addresses for the communication core program are the arguments. The communication core program in RAM is used for communication. Make sure that you copy the core program with the function **irRamCopy** before calling this function.

void **irMountThumbCore**(void)

The communication core program in the ROM is used (ROM version) for the communication after calling this function.

void **irInit**(void)

Initializes the communication status. Please call this function once before each data transfer.

u8 **irSendConnect**( u16 bps)

Sends a connection request as the transmitting side to the other machine.
Using the argument, **bps**, determine the communication speed. (See "1.2 Communication Speed".) Sets up the registers IE, IME, timer control, and the setting values. After the connection request has been sent, this function returns IR_SUCCESS if the receiving side sends a normal connection reply. Otherwise, IR_FAILURE is returned.

u8 **irRecvConnect**( u16 bps)

Sends a connection reply as the receiving side to the other machine. Using the argument, **bps**, determine the communication speed. (See "1.2 Communication Speed") Sets up the registers IE, IME, timer control, and the setting values.

If there was a normal connection request, this function sends a connection reply and returns IR_SUCCESS. Otherwise, IR_FAILURE is returned.

u8 **irSendPacket**( u16* addr, u8 halfwords)

This function sends the header + data + checksum and then receives the status.
Using the argument, **addr**, specify the send data starting address and set the transmission data number in half-words.

If communication proceeds normally, IR_SUCCESS is returned. Otherwise, IR_FAILURE is returned. Up to 255 half-words can be transmitted with one call of this function. If more data is needed, this function must be called more than once.

u8 **irRecvPacket**( u16* addr, u8 half-words)

This function receives the header + data + checksum and then sends the status. Using the argument, **addr**, specify the receive data starting address and set the transmission data number in half-words.

If communication proceeds normally, IR_SUCCESS is returned. Otherwise, IR_FAILURE is returned. Up to 255 half-words can be received with one call of this function. If more data is needed, this function must be called more than once.

void **irDisconnect**( void)

This function terminates IR communication while on standby. Restore the registers IE, IME, timer control, and setting values from the buffer which are saved when the function irBegin is called.

Communication is restarted with irSendConnect or irRecvConnect. If you are interrupting communication for a long time, use the irEnd function in order to reduce the power consumption.

void **irEnd**( void)

Call this function at the end of the IR communication. Stop the timer. Returns the content of IE, IME, timer control and the setting values, which are saved when the function irBegin is called. Also, this function operates the R register and shuts down the IR adapter.

### irResult structure

This structure stores various kinds of information for IR communication. Please keep it in the CPU internal RAM. The state of communication can be obtained by referencing this information.
The communication library uses this structure, so be careful not to write to the structure's members. This may disrupt normal communication.

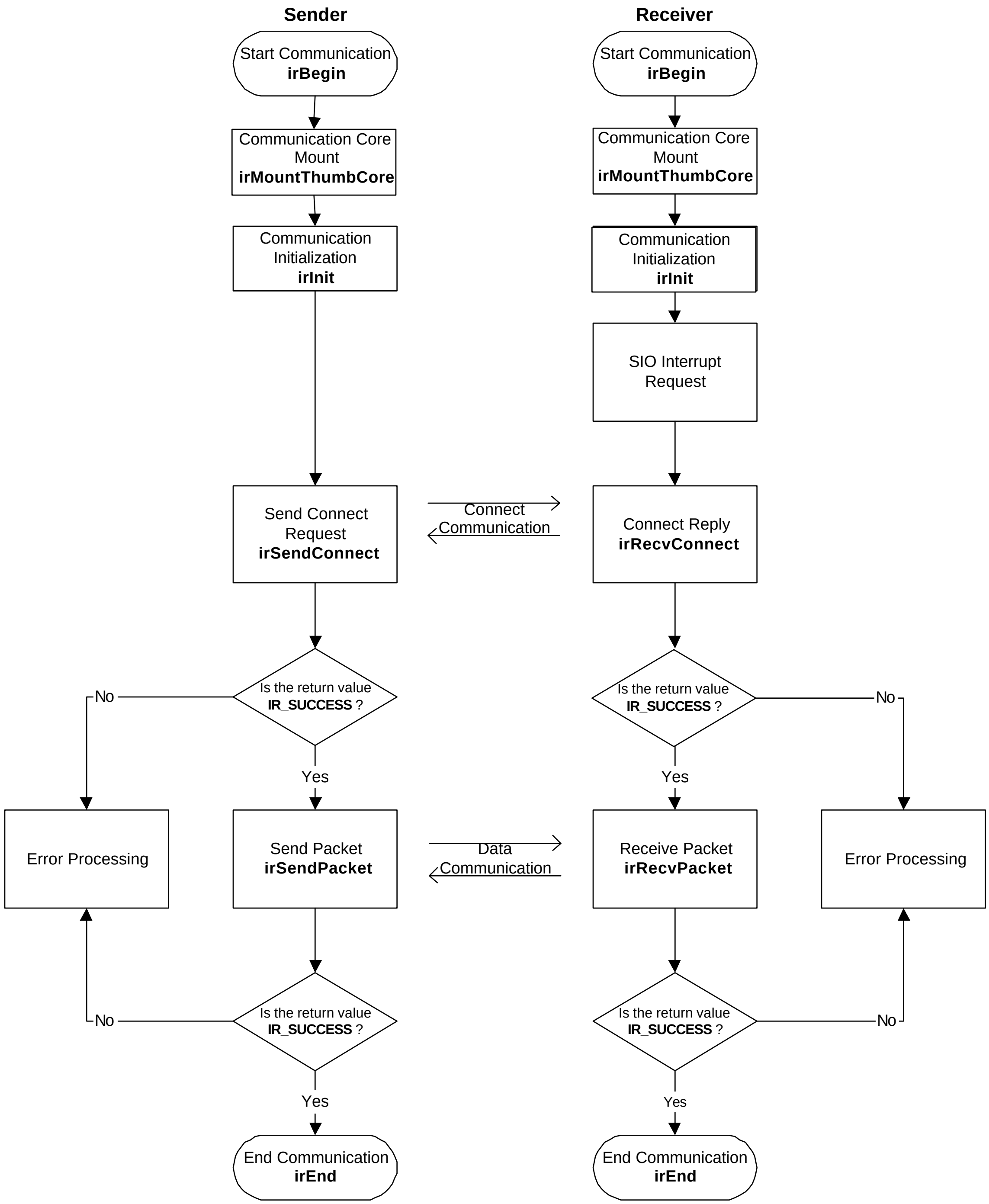
```
typedef   struct    {
          u8        halfwords;
          u8        mode;
          u16       status;
          u16       sum;

}irResult;
```

halfwords: Contains the number of bytes being transmitted or received.
mode: Contains the constant that indicates whether the connection is to the transmitting side or to the receiving side.
status: Contains the communication status.
sum: Contains the checksum value.

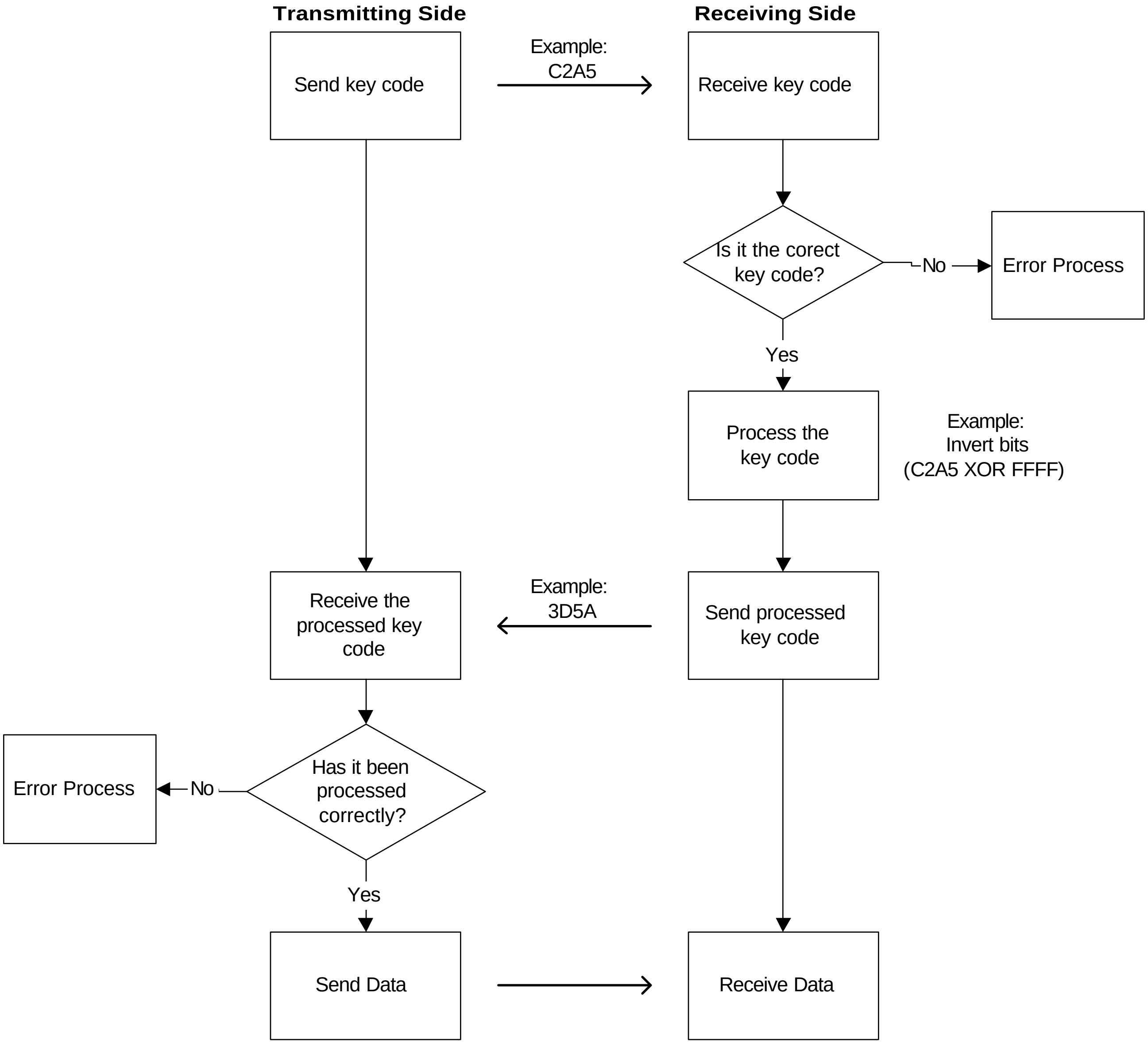
## 3.2 Flowcharts

The most basic communication procedure is shown below:


Sender
Receiver
Start Communication **irBegin**
Start Communication **irBegin**
Communication Core Mount **irMountThumbCore**
Communication Core Mount **irMountThumbCore**
Communication Initialization **irInit**
Communication Initialization **irInit**
SIO Interrupt Request
Send Connect Request **irSendConnect**
Connect Communication
Connect Reply **irRecvConnect**
Is the return value **IR_SUCCESS** ?
No
Yes
Is the return value **IR_SUCCESS** ?
No
Yes
Error Processing
Send Packet **irSendPacket**
Data Communication
Receive Packet **irRecvPacket**
Error Processing
Is the return value **IR_SUCCESS** ?
No
Yes
Is the return value **IR_SUCCESS** ?
No
Yes
End Communication **irEnd**
End Communication **irEnd**

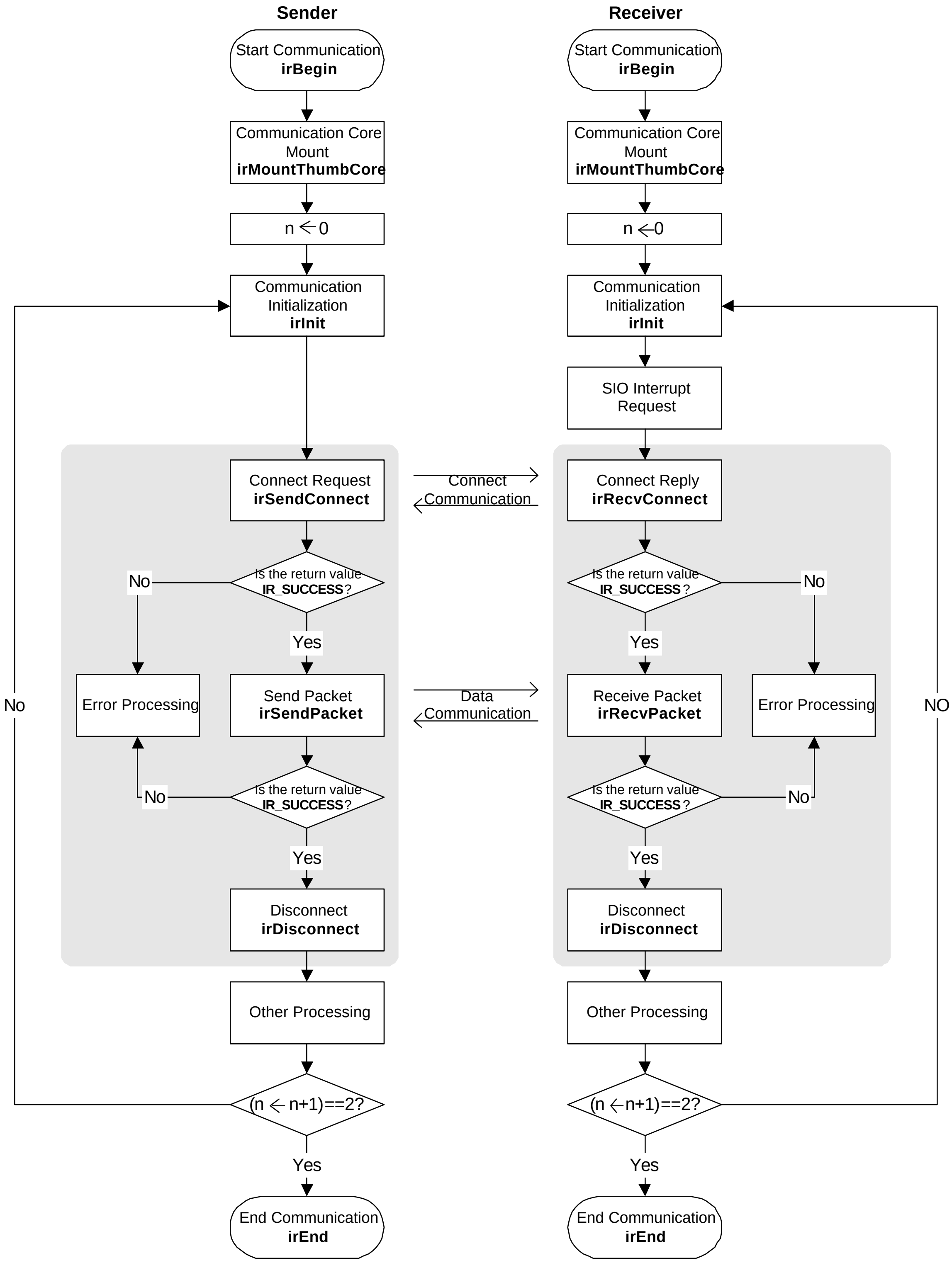
When using the RAM to carry out high speed communication, replace irMountThumbCore with irRamCopy and irMountArmCore.

In actual communication, the transfer of data does not start immediately upon connection, but only after an exchange of key codes and other procedures. This is done in order to prevent erroneous communication between different games. See the flowchart shown below:



The key code processing can involve such procedures as bit inversion, swapping upper and lower 8bits, doing an arithmetic operation or a logic operation with a certain constant, or performing a combination of these.

It is possible to temporarily halt communication using irDisconnect to perform another process. Don't perform another process until irSendConnect has completed. The flowchart below shows this process:



## 4.0 Cautions

Please note the following when using the IR communication library provided by Nintendo:

- Use after turning off the prefetch buffer. The prefetch buffer must be disabled before using.

- The communication speed is specified depending on the following conditions:

  * The location of the data buffers for sending and receiving data
  * Cartridge ROM wait cycles

  Be sure to transfer data at a communication speed that conforms to the conditions. Even if you are able to transfer data to your AGB (for development use) at speeds that are faster than the specified rate, it may not be possible to perform the same communication on the commercial AGBs, which show larger variations among machines.

- Before communicating, make sure the following conditions for the sender and receiver agree with each other.

  *Communication Speed
  *Game Pak ROM wait cycles
  *Use either the ROM version or the RAM version as the communication core.
  *Data Buffer Location (CPU Internal RAM or CPU External RAM)

- If you discover any bugs or problems, please contact the Software Development Support Group at support@noa.com.

Regarding the CPU internal RAM version, pay attention to the following items:

- With the version that operates in the CPU internal RAM, the ARM-coded communication core program is copied inside the CPU Internal RAM for fast no-wait operations. This requires a buffer for storing the communication core program in CPU Internal RAM.

- Be sure to set aside a buffer of the specified size.
  -- If the capacity is insufficient, the program will hang.

- Set the starting address of the program buffer to a multiple of 4. (32bit alignment
  -- By setting the address to a multiple of 4, the CPU will operate in ARM mode.
  If the address is not set to a multiple of 4, the program will execute as Thumb code and hang.

- Be sure to set aside the program buffer in the CPU internal RAM.
  -- Execution of the ARM code requires a 32bit bus of CPU internal RAM.
  The CPU external RAM has a 16bit bus, so the program will hang.
  Also, the communication speed is specified assuming that it is a nowait CPU internal RAM.

- Make sure you allocate the library BSS area to CPU internal RAM.

### Notes for creating your own drivers for IR Communication

- Set aside turnaround time as determined by IrDA for at least 0.5ms. Here, "turnaround time" refers to the time when the Ir module is temporarily not receiving light signals because the light transmitted by the module has entered the receiver side of the same module.
- At least 2ms are required before the Ir module can start up from shutdown mode. Make sure that the AGB verifies this and secures enough time. The Ir module is available for use when High is input to the SD terminal and Low is input to the SC terminal. In any other state, the module is in shutdown mode.
- Set aside a period of at least 1.4 µs for High output to the SO terminal. (Needs to be verified.) If the period is shorter, the other machine may not be able to sense the received light. However, try to avoid a long period of time since this will greatly increase the power consumption.
- AGB's crystal resonator operates within a certain range of error (due to individual differences and to temperature variations). Please take this into consideration for high-speed protocols that reliability depends on the timer. (Synchronize periodically) The maximum difference is approximately +/-80ppm. (Need to confirm).
- In order to reduce power consumption, please set the Ir module to shutdown mode when not performing IR communication.
- Even if communication is interrupted due to shielding or external light, make sure you carry out error processing so there are no problems in operation.
- Be sure to perform sufficient tests such as aging and others to verify the reliability of the system.