Renesas Synergy™

R30AN0321JJ0100
Rev.1. 00
2017.12.28

NetX を使ったソケット通信の応用例：マルチクライアントサーバ

要旨

本書では、NetX を使ったソケット通信について説明します。クライアント動作、及び複数のクライアントと通信を行うマルチクライアントサーバ間の通信を例としています。また、MAC アドレスの変更方法についても説明します。

また本書に付属のサンプルプログラムは、表 1 の環境で動作します。

表 1　動作環境

Renesas Synergy™ Software Package (SSP)	v1.3.2
e2studio ISDE	v5.4.0.023
Renesas Synergy™ Standalone Configurator (SSC)	v5.4.0.023
IAR Embedded Workbench® for Renesas Synergy™ (IAR EW for Synergy)	v7.71.3
Renesas Flash Programmer	v3.03.01

動作確認デバイス

SK-S7G2 v3.1

目次

1. はじめに

1.1 概要

本書では、NetX を使用したマルチクライアントサーバの実装例を下に、NetX を使ったソケット通信及び MAC アドレスの変更方法について説明します。

本書付属のサンプルプログラムは、マルチクライアントサーバ - クライアント間でソケット通信を行います。マルチクライアントサーバは、クライアントの識別やプッシュスイッチ押下によるパケット送信対象の切り換え及び、LED をトグルするコマンドの送信によって対象クライアントの LED の点灯／消灯を切り換えます。また、クライアントについては、マルチクライアントサーバと同様にプッシュスイッチ押下によって LED をトグルするコマンドを送信し、マルチクライアントサーバの LED の点灯／消灯を切り換えます。また、各デバイスは、電源投入後の初期化時に、データフラッシュに格納された情報を基にして自局 MAC アドレスを設定します。

1.2 参考文献

[1] Renesas, “NetX™ User Guide (R11UM0004EU0590 Rev. 5.90)”.

[2] Renesas, “NetX™ Dynamic Host Configuration Protocol for Clients User’ Manual (R11UM0014EU0591 Rev. 5.91)”.

[3] Renesas, “ThreadX® User’s Manual: Software (R11UM0006EU0500 Rev. 5.00)”

[4] Renesas, “S7G2 User’s Manual: Microcontrollers (R01UM0001EU0120 Rev. 1.20)”

[5] Renesas, “Starter Kit SK-S7G2 User’s Manual (R12UM0004EU0100 Rev. 1.00)

[6] Renesas, “Renesas Flash Programmer V3.03 フラッシュ書き込みソフトウェア ユーザーズマニュアル (R20UT4066JJ0100 Rev. 1.00)

2. サンプルプログラム仕様

図 1 にシステム構成を示します。また、マルチクライアントサーバ、クライアントの動作概要は以下の通りです。また、マルチクライアントサーバ、クライアントともに SK-S7G2 上で動作します。S7G2, SK-S7G2 については参考文献[4][5]を参照してください。

- マルチクライアントサーバ
  - IP アドレス 192.168.10.5／ポート番号 65535 で クライアントからの接続待ち状態に入り、その後 LED1,2,3 を消灯する。また接続があった場合、クライアント A、もしくはクライアント B かを識別する。
  - S5 を押すことで、LED トグルコマンドを送信するクライアントを切り換える。また LED1,2 にその状態を表示する（図 2）。
  - S4 を押すことで、対象のクライアントに対して LED トグルコマンドを送信する。また、S4 を押した回数が 5 回になると、通信中の全てのクライアントとの通信を切断する。
  - クライアントから LED トグルコマンドを受信した際は、LED3 をトグルする。
  - 全てのクライアントとの通信が終了した場合は、LED1,2,3 を点灯する。
- クライアント
  - 電源投入後、DHCP による IP アドレス取得を行い、マルチクライアントサーバへ接続する。接続が成功した場合は LED1 を点灯させる。
  - マルチクライアントサーバから LED トグルコマンドを受信した場合は LED3 をトグルさせる。
  - S4 を押すことで、LED トグルコマンドを マルチクライアントサーバへ送信する。また、S4 を押した回数が 5 回になると、マルチクライアントサーバとの通信を切断する。
  - Server との通信が終了した場合は、LED1 を消灯する。
- 共通機能
  - 電源投入後にデータフラッシュから MAC アドレス（シリアル ID に対応する第 5,6 オクテットの情報）を読み取り、自局 MAC アドレスとして設定する。

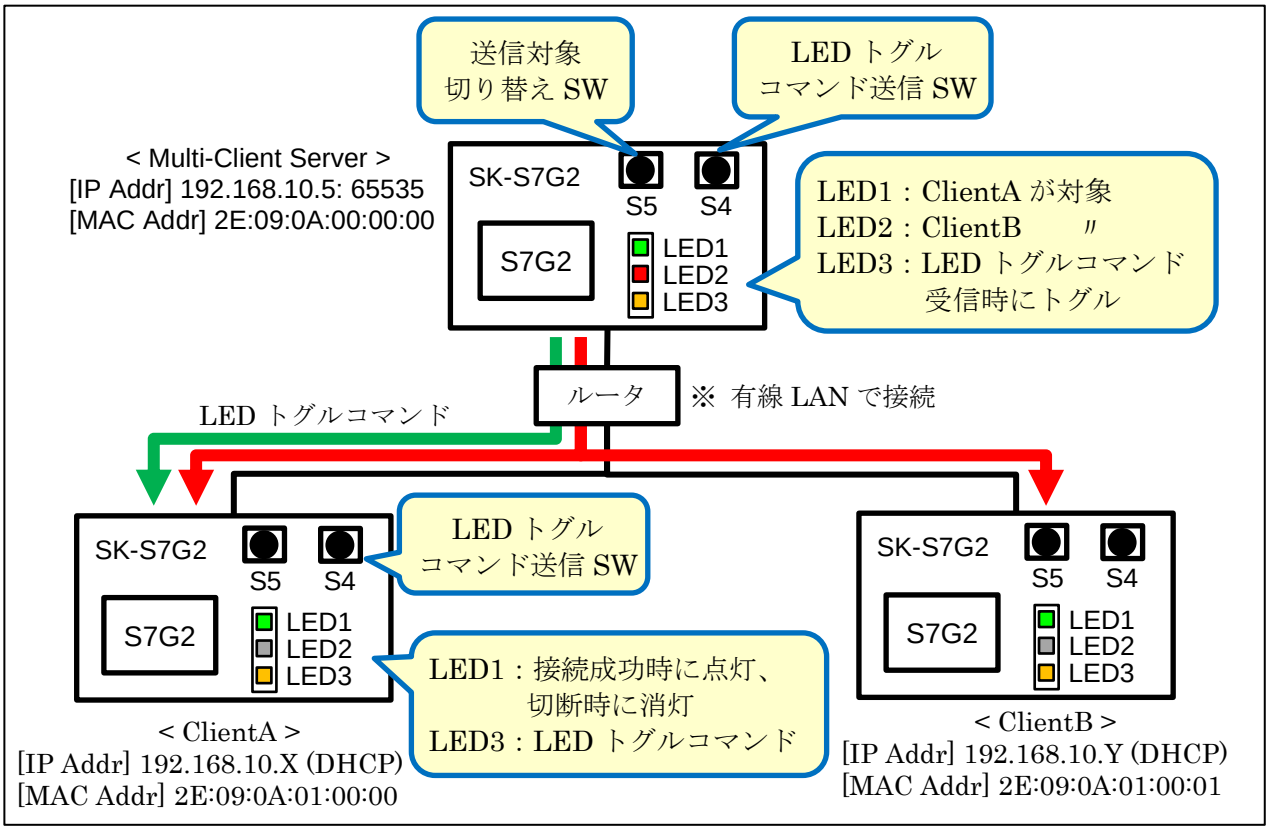

図 1　システム構成

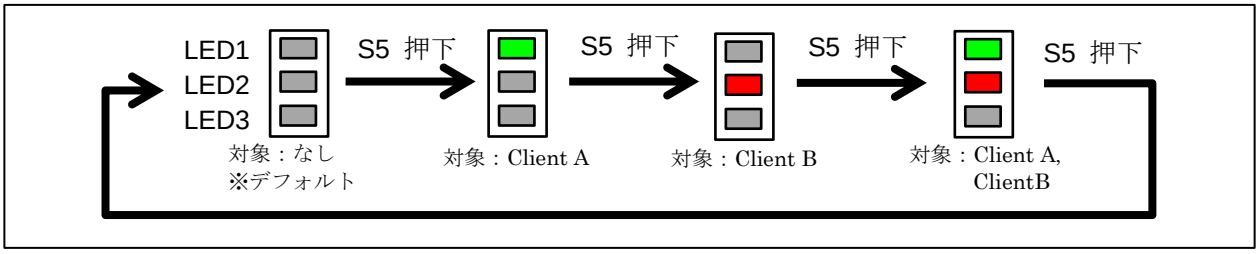

図 2　マルチクライアントサーバ上の LED1,2 と送信対象の関係

3.　ハードウエア構成

3.1　ブロック図

マルチクライアントサーバ、クライアントのブロック図をそれぞれ図 3,4 に示します。両者とも Ethernet PHY, S4, LED1,3 を使用し、さらにマルチクライアントサーバのみ S5, LED2 も使用します。

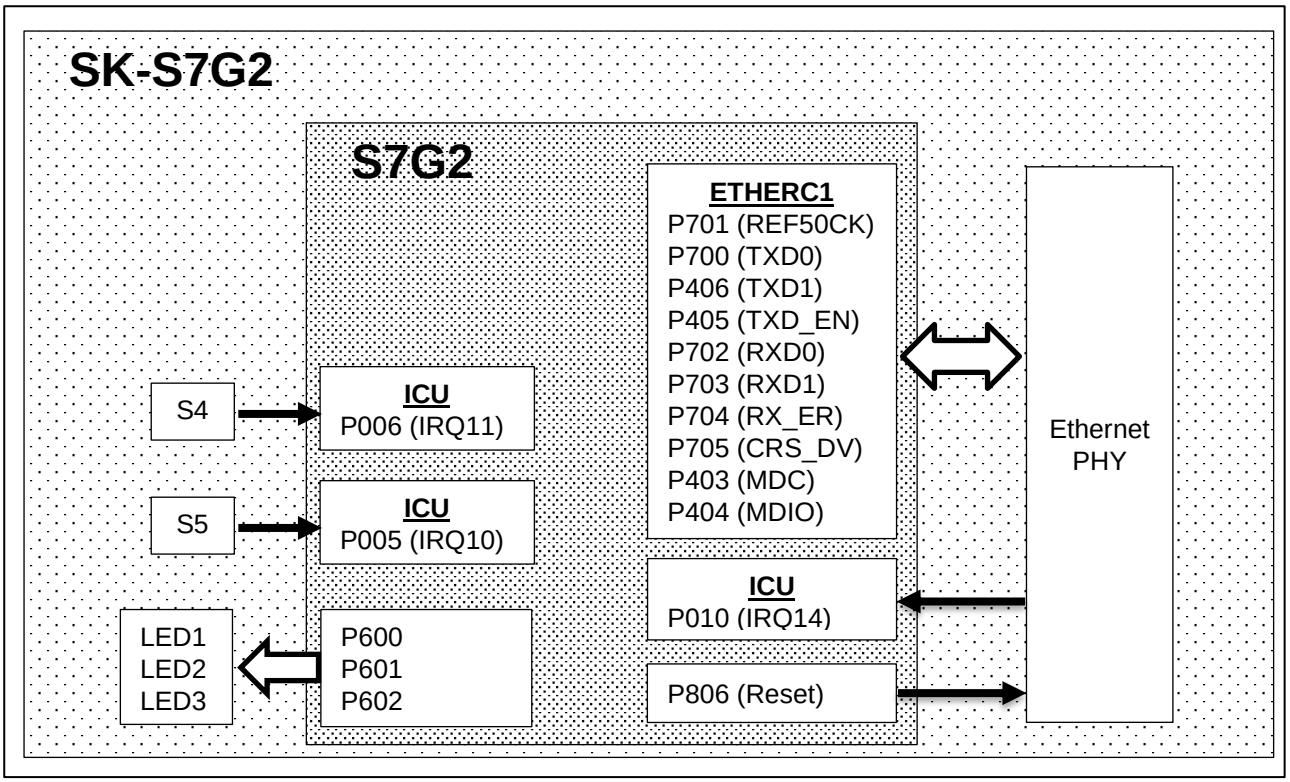

図 3　マルチクライアントサーバのブロック図

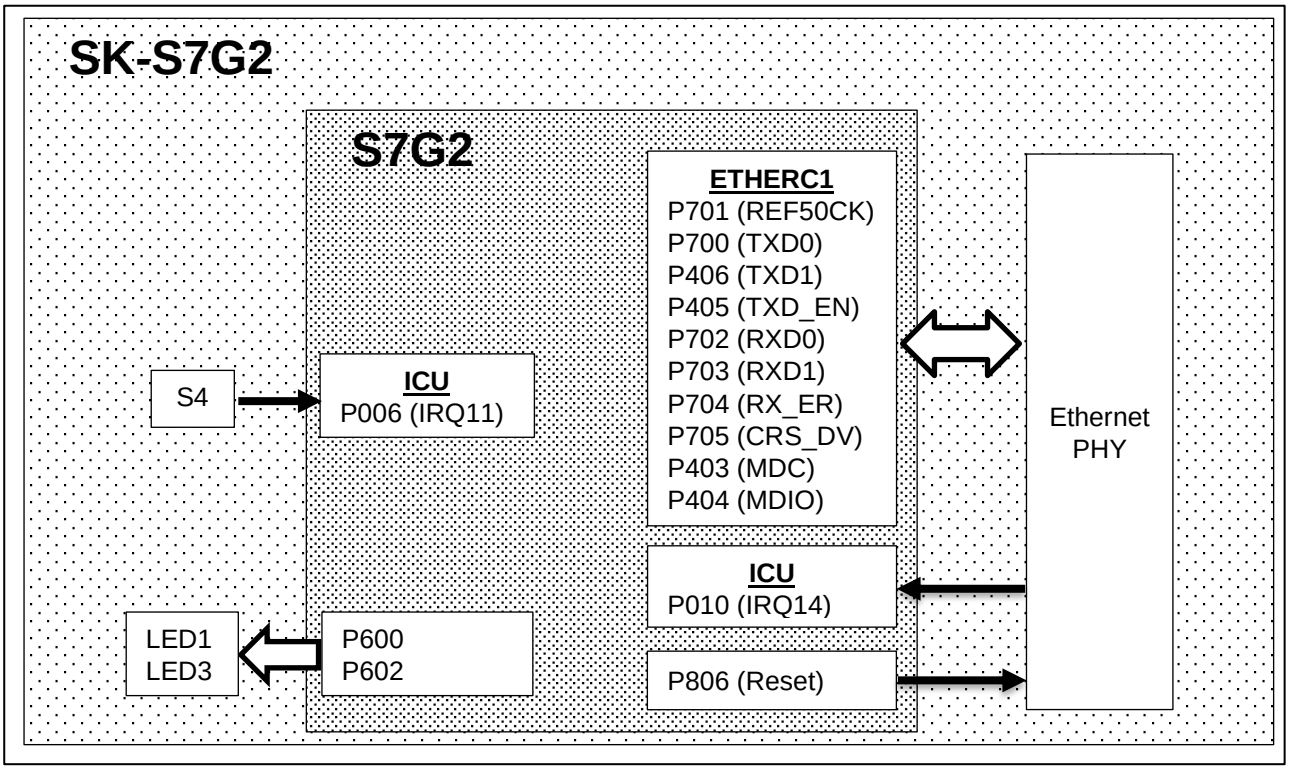

図 4　クライアントのブロック図

4.　ソフトウエア構成

4.1　ソフトウエアのインストール

付属の”Synergy Project Import Guide” に従い、以下のプロジェクトをインポートしてください。

・マルチクライアントサーバ用プロジェクト：multi_client_server.zip
・クライアント用プロジェクト：client.zip

4.2　モジュール構成

本書に付属のサンプルプログラムで使用する主な SSP モジュールを表 2, 3 に示します。

表 2　マルチクライアントサーバが使用する SSP モジュール

モジュール種別	モジュール名	用途
X-ware	tx	ThreadX®：スレッドの制御など
	nx	NetX™：TCP/IP スタックの制御
HAL Driver	r_ioport	LED1,2,3 の制御
Framework	sf_external_irq	S4, S5 の制御

表 3　クライアントが使用する SSP モジュール

モジュール種別	モジュール名	用途
X-ware	tx	ThreadX®：スレッドの制御など
	nx	NetX™：TCP/IP スタックの制御
	nx_dhcp	NetX™ Dynamic Host Configuration Protocol for Clients：DHCP による IP アドレス取得用
HAL Driver	r_ioport	LED1,3 の制御
Framework	sf_external_irq	S4 の制御

4.3　マルチクライアントサーバ

マルチクライアントサーバを実現するための NetX 及び ThreadX が提供する API の使い方（シーケンス）について説明します。NetX, ThreadX に関する詳細は参考文献[1][3]を参照ください。

4.3.1　初期化＆リッスン開始時のシーケンス

複数のクライアントとソケット通信を行うための初期化、及びリッスン時のシーケンスは図 6 の通りです。また対応する関数は multi_client_server/src/multi_client_server/multi_client_server.c 内の mcs_open()です。

シーケンスの概要を以下に示します。※①～③は許容するクライアント数分繰り返す

① tx_queue_create()をコールし、受信パケット用のキューを作成する。
⇒ これはクライアントから受信したデータを指定バイト数取得するためのものであり、受信したパケットは、一旦このキューへ格納されます。詳細は③を参照ください。

② nx_tcp_socket_create()をコールし、ソケットを作成する。また、引数に切断要求受信時のコールバック関数を登録する。
⇒ 切断要求受信時のコールバック関数の概要は以下の通りです。切断要求の送信と複数のクライアントとソケット通信を行うため、再度リッスンを行います。
※以下の処理は、本コールバック関数内で別途生成したスレッド内で実施します。この理由は、下記でコールする NetX API にタイムアウト値（NX_NO_WAIT 以外）を指定するためです。本コールバック関数は NetX の内部スレッド上でコールされますが、内部スレッド上ではタイムアウトを待つことができません（NX_NO_WAIT 以外を指定しても NX_NO_WAIT と同じ動作になります）。このため、サンプルプログラムでは、NetX の内部スレッドとは別のスレッドを生成し、その中で処理を行う実装としています。
・受信したパケットが RST ではなかった場合、nx_tcp_socket_disconnect()をコールして切断要求を送信する。
・nx_tcp_server_socket_unaccept()をコールし、切断要求を受信したソケットとポートの関連付けを解除する。

・すでにリッスン中のソケットが無い場合、接続確立していない空きのソケットに対して nx_tcp_server_socket_relisten()をコールし、再度リッスンを実施する。
・ユーザ指定のコールバックを呼び出す。この際、切断要求のあったソケットに対応するビットを引数として渡す。

③　nx_tcp_socket_receive_notify()をコールし、作成したソケットに対してパケット受信時のコールバック関数を登録する。
⇒ パケット受信時のコールバック関数の概要は以下の通りです。本書付属のサンプルプログラムでは、パケット受信を契機にパケットを取得し以下のようにキューへ格納する動作としていることから nx_tcp_socket_receive_nofity()を使用していますが、nx_tcp_packet_receive()にタイムアウト値（NX_NO_WAIT 以外）を指定することで、nx_tcp_packet_receive()のみで受信パケットの到着を待つことも可能です。
・nx_tcp_socket_receive()をコールし、受信したパケットを取得する。以降の処理を受信したパケットを全て取得するまで繰り返す。
・tx_queue_send()をコールし、パケット内のデータを取り出す（1 バイト）。データは、受信したソケットに対応するキューへ格納する。これをパケットに含まれるデータ数分繰り返す。
・nx_packet_release()をコールし、パケットを解放する。
※ nx_tcp_socket_receive()に成功した場合ユーザアプリ側でパケットを解放する必要があります。

④　nx_tcp_server_socket_listen()をコールし、1 番目に作成しソケットに対してリッスンを開始する。また、引数に接続要求受信時のコールバック関数を登録する。
⇒ 接続要求受信時のコールバック関数の概要は以下の通りです。要求受け入れ後、クライアント情報を更新し、さらに複数のクライアントからの接続要求を受け付けるため再度リッスンを行います。
※以下の処理は切断要求受信時のコールバック関数同様、別途生成したスレッド内で実施します。
・nx_tcp_server_socket_accept()をコールし、接続要求を受け入れる。
・クライアント情報を更新（接続要求を受信したソケットに対応するビットをセット）
・通信未確立のソケットに対して nx_tcp_server_socket_relisten()をコールし再リッスンを実施。
・ユーザ指定のコールバック関数を呼び出す。この際、上記でセットしたビット（接続要求があったクライアントに対応）を引数として渡す。

4.3.2　パケット送信時のシーケンス

クライアントへパケットを送信するためのシーケンスは図 6 の通りです。

また、対応する関数は、multi_client_server/src/multi_client_server/multi_client_server.c 内の mcs_send()です。本関数は、引数に指定されたビットに対応するクライアント分、送信データを複製してそれぞれのクライアントへパケットを送信します。

シーケンスの概要を以下に示します。※①～③は指定されたクライアント数分繰り返す

①　nx_packet_allocate()をコールし、送信パケット用のメモリを確保する。

②　nx_packet_data_append()をコールし、送信データを設定する。

③　nx_tcp_socket_send()をコールし、パケットを送信する。

※パケット受信時(nx_tcp_socket_receive())とは異なり、送信が成功した場合は内部でパケットが解放されます。ただし、送信に失敗した場合は内部で解放されないため、ユーザアプリ側で nx_packet_release()をコールしてパケットを解放する必要があります。

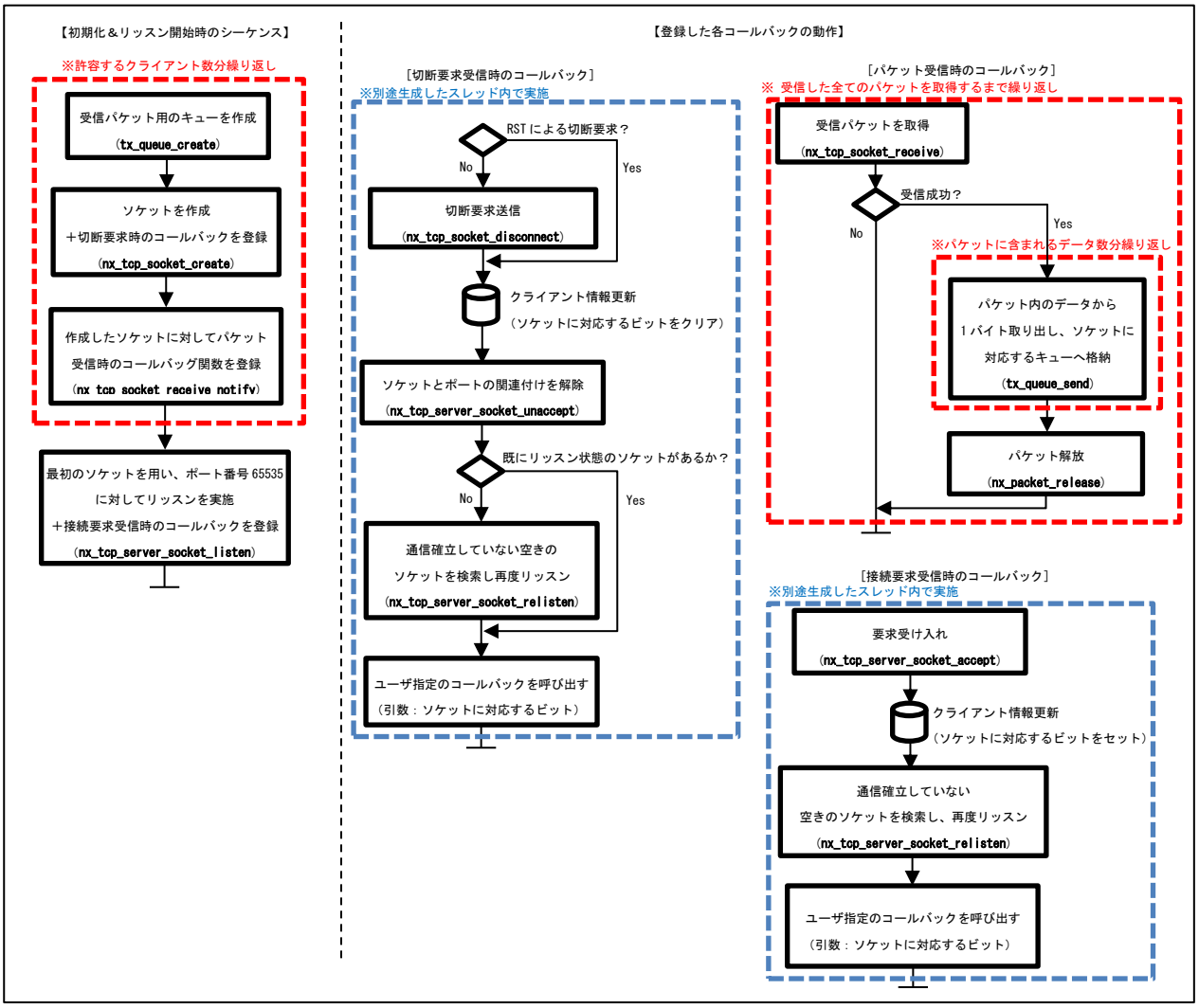

図 5　初期化＆リッスン開始時のシーケンス

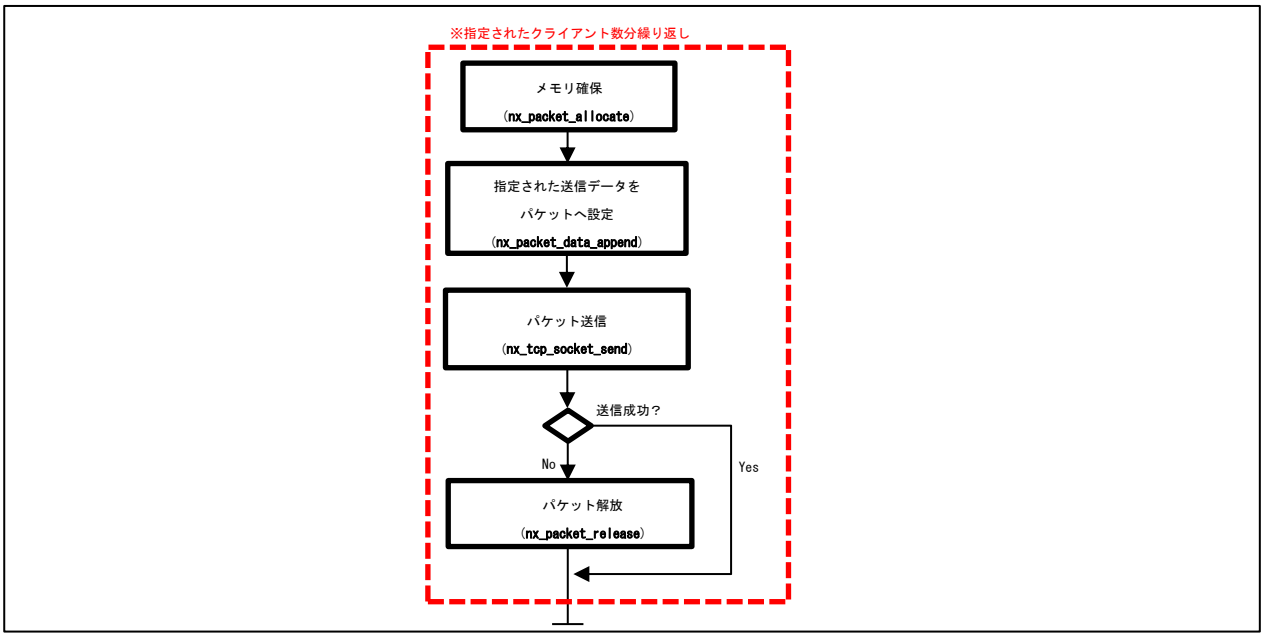

図 6　パケット送信時のシーケンス

4.3.3　受信パケット取得時のシーケンス

キューに格納された受信パケットから指定バイト数分のデータを取得するためのシーケンスは図 7 の通りです。また、対応する関数は、multi_client_server/src/multi_client_server/multi_client_server.c 内の mcs_receive()です。本関数では、引数に指定されたビットに対応する 1 つのクライアントから受信したパケットを取得します。

シーケンスの概要を以下に示します。※①は指定バイト数のデータを取得できるまで繰り返す

① tx_queue_receive()をコールし、指定されたクライアントに対応するキューから 1 バイト分のデータを取得する。

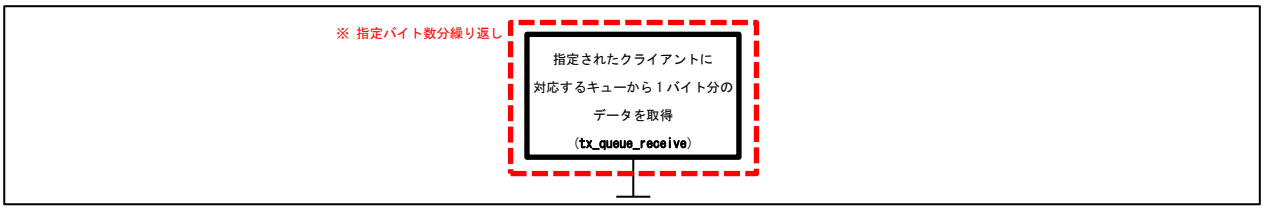

図 7　受信パケット取得時のシーケンス

4.3.4　通信終了時のシーケンス

クライアントとの通信を終了するためのシーケンスは図 8 の通りです。また、対応する関数は、multi_client_server/src/multi_client_server/multi_client_server.c 内の mcs_close()です。本関数は、（通信確立中の）クライアント全てに対して切断要求の送信と、初期化時に生成したリソースの解放を行います。

シーケンスの概要を以下に示します。

※①～④を許容するクライアント数分繰り返す。

① nx_tcp_socket_disconnect()をコールし、切断要求を送信する。
※対象のソケットが通信確立中でなければ、本関数をコールしても切断要求は送信されません。

② nx_tcp_server_socket_unaccept()をコールし、ソケットとポートの関連付けを解除する。

③ nx_tcp_socket_delete()をコールし、ソケットを削除する。

④ tx_queue_delete()をコールし、受信パケット用のキューを削除する。

⑤ nx_tcp_server_unlisten()をコールし、リッスンを解除する。

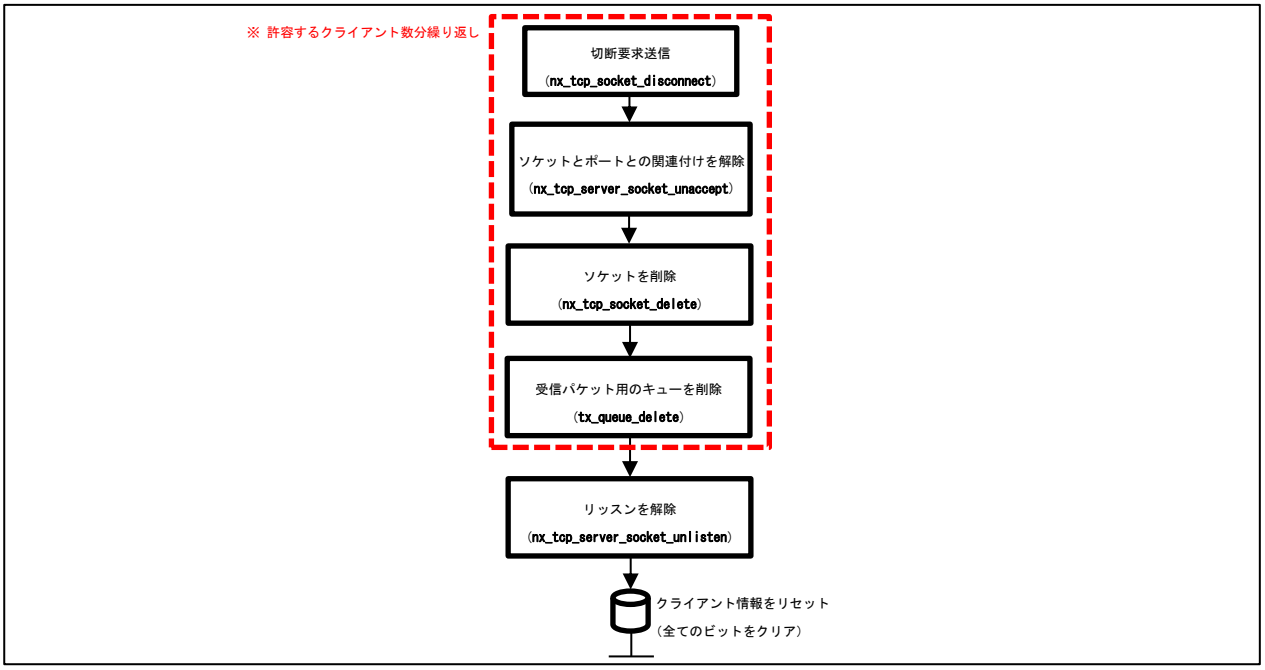

図 8　通信終了時のシーケンス

4.3.5　アプリのシーケンス

アプリは、先述の関数（mcs_open(), mcs_send(), mcs_receive(), mcs_close()）を組み合わせることで、2 章で説明したマルチクライアントサーバの動作を実現しています。

まず初期化時の動作は図 9 の通りです。App Thread は最初に mcs_open()をコールし、初期化及びリッスンを開始し、その後 LED1,2,3 を消灯します。クライアントから接続要求があった場合、mcs_open()で登録したコールバックが呼び出され、クライアント識別のためにクライアント名を取得します。取得したクライアント名とコールバックの引数で渡されたビットを紐づけて管理します。また、その後対象のクライアントからのパケット受信を行うための Sub Thread を生成します。これらの処理をクライアント A、B が接続するまで繰り返します。

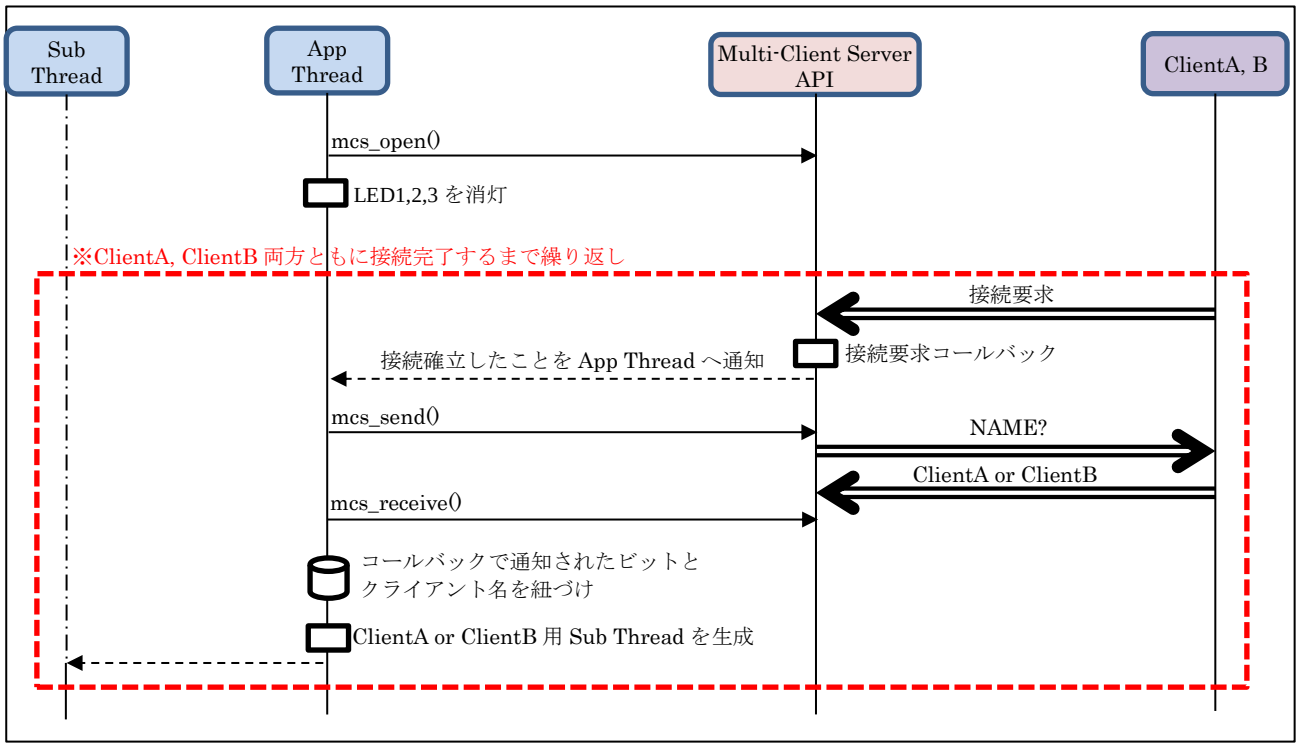

図 9　初期化時のシーケンス

初期化後の App Thread の動作は以下の通りです。S5 が押される度に送信対象を図 2 のように切り替えます。また、S4 が押される度に送信対象のクライアントに対応するビットを取得し mcs_send()に指定することで、LED トグルコマンドを送信します。

また、S4 を押した回数が 5 回を超えた場合もしくは、全てのクライアントとの通信が終了した場合、mcs_close()をコールして、通信中のクライアントに対して切断要求を送信します（既に全てのクライアントとの通信が終了している場合、mcs_close()をコールしても切断要求は送信せず、mcs_open()時に生成したリソースの解放のみ実施）。そして最後に、LED1,2,3 を点灯します。

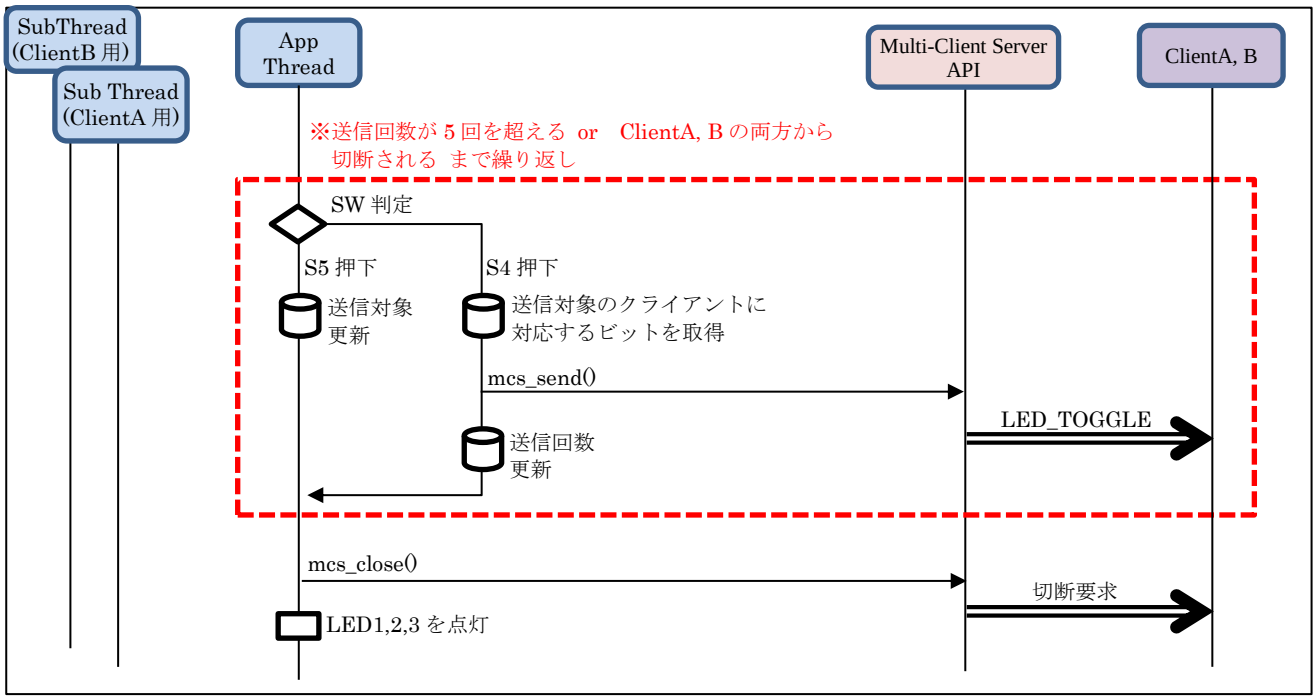

図 10　App Thread の動作

また、初期化時に生成した Sub Thread の動作は図 11 の通りです。Sub Thread は mcs_receive()をコールして、対象のクライアントからの LED トグルコマンドを待ちます。受信後は LED3 をトグルします。また、対象のクライアントから切断要求があった場合は、Sub Thread を終了します。

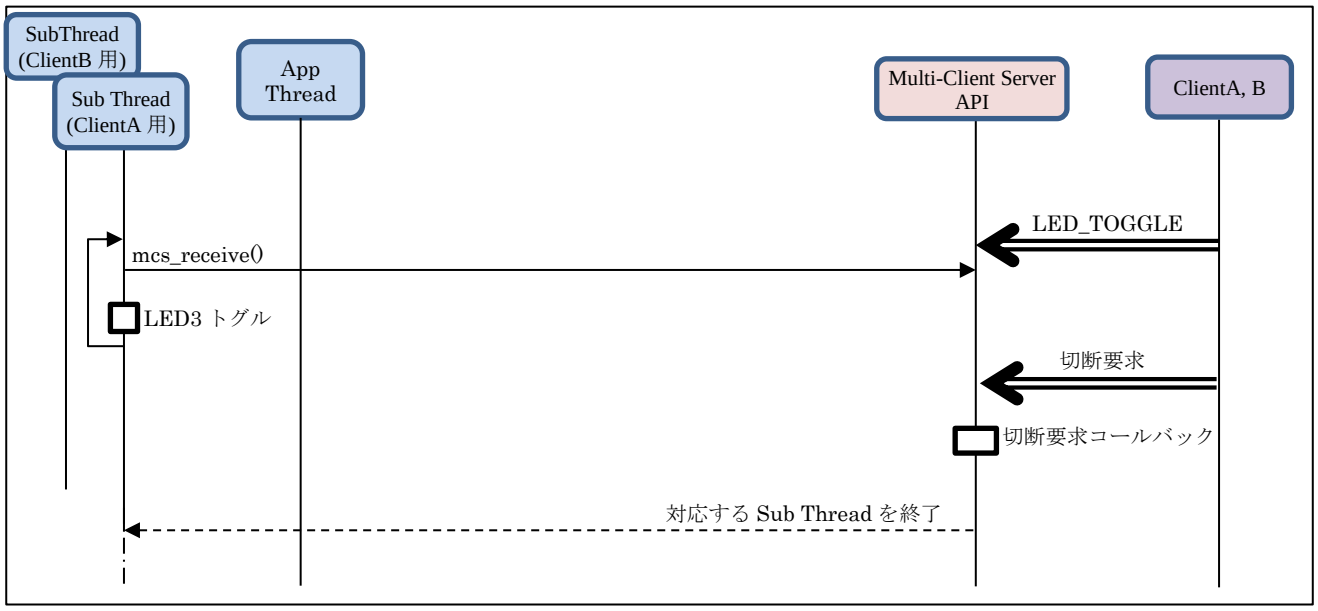

図 11　Sub Thread の動作

4.4　クライアント

クライアントを実現するための NetX 及び ThreadX が提供する API の使い方（シーケンス）について説明します。NetX 及び ThreadX に関する詳細は参考文献[1][3]を参照ください。また、NetX DHCP については説明を割愛しています、詳細は参考文献[2]を参照ください。

4.4.1　初期化＆サーバ接続時のシーケンス

初期化、及びサーバ接続時のシーケンスは図 12 の通りです。また、対応する関数は、client/src/client/client.c 内の client_open()です。

シーケンスの概要を以下に示します。

① DHCP によって IP アドレスを取得する。
② tx_queue_create()をコールし、受信用のキューを作成する。
　⇒ マルチクライアントサーバと同様、サーバから受信したデータを指定バイト数取得するためのものです。
③ nx_tcp_socket_create()をコールし、ソケットを作成する。また、引数に切断要求受信時のコールバック関数を登録する。
　⇒ 切断要求受信時のコールバック関数の概要は以下の通りです。
　　※4.3.1 項 ④で説明したマルチクライアントサーバの切断要求受信時と同様に、以下の処理は別途生成したスレッド内で実施されます。
　　・受信したパケットが RST ではなかった場合、nx_tcp_socket_disconnect()をコールして切断要求を送信する。
　　・ユーザ指定のコールバックを呼び出す。
④ nx_tcp_socket_receive_notify()をコールし、作成したソケットに対してパケット受信時のコールバック関数を登録する。
　⇒ パケット受信時のコールバック関数の概要は以下の通りです。
　　・nx_tcp_socket_receive()をコールし、受信したパケットを取得する。以降の処理を受信したパケットを全て取得するまで繰り返す。
　　・tx_queue_send()をコールし、パケット内のデータを取り出す（1 バイト）。データは受信したソケットに対応するキューへ格納し、これをパケットに含まれるデータ数分繰り返す。
　　・nx_packet_release()をコールし、パケットを解放する。
⑤ nx_tcp_client_socket_bind()をコールし、作成したソケットをバインドする。この際、第 2 引数に”NX_ANY_PORT”を設定し空いているポート番号を使用する（以降、client_*()関数はこのポート番号を使用してソケットを行います）。
⑥ nx_tcp_client_socket_connect()をコールし、マルチクライアントサーバへ接続する。

4.4.2　パケット送信時のシーケンス

クライアントへパケットを送信するためのシーケンスは図 13 の通りです。また、対応する関数は client/src/client/client.c 内の client_send()です。シーケンスの概要を以下に示します。

① nx_packet_allocate()をコールし、送信パケット用のメモリを確保する。
② nx_packet_data_append()をコールし、送信データを設定する。
③ nx_tcp_socket_send()をコールし、パケットを送信する。

4.4.3　受信パケット取得時のシーケンス

キューに格納された受信パケットから指定バイト数分のデータを取得するためのシーケンスは図 14 の通りです。また、対応する関数は、client/src/client/client.c 内の client_receive()です。

シーケンスの概要を以下に示します。※①は指定バイト数のデータを取得できるまで繰り返す

① tx_queue_receive()をコールし、指定されたクライアントに対応するキューから 1 バイト分のデータを取得する。

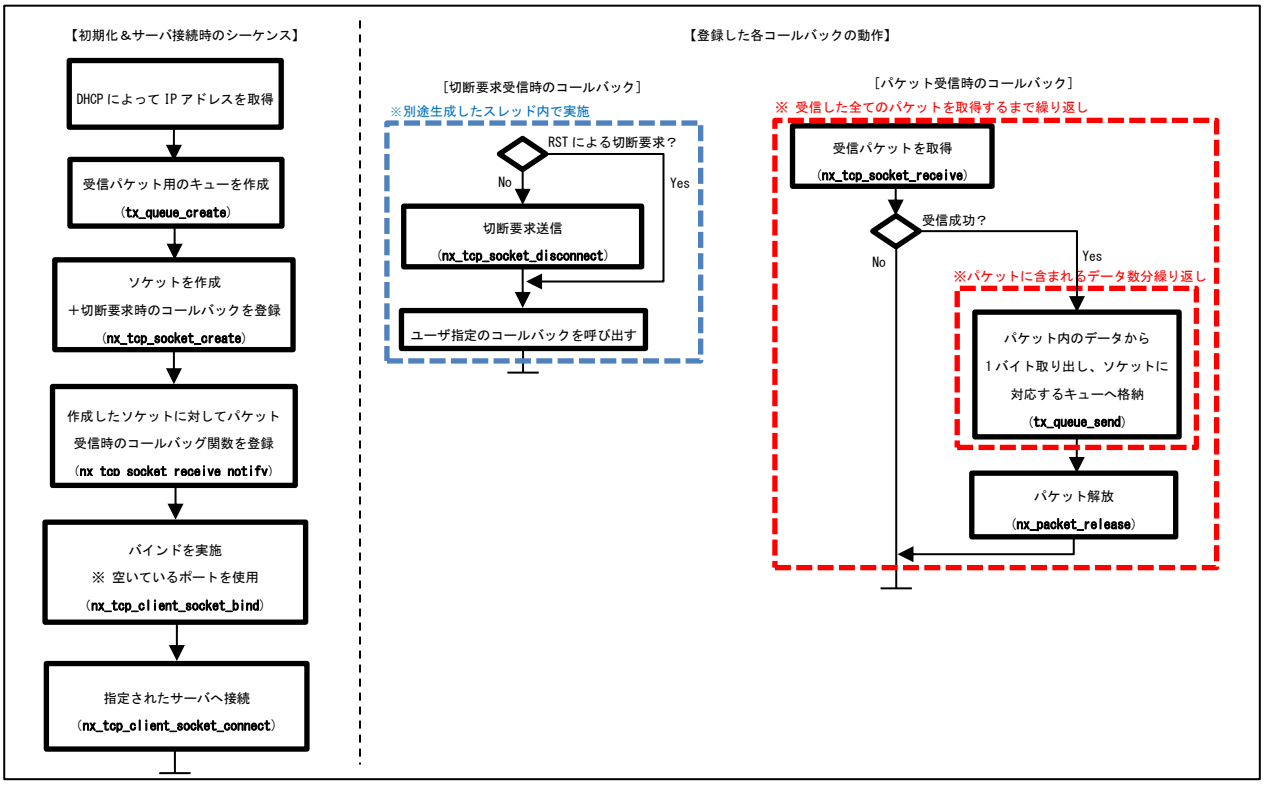

図 12　初期化＆サーバ接続時のシーケンス

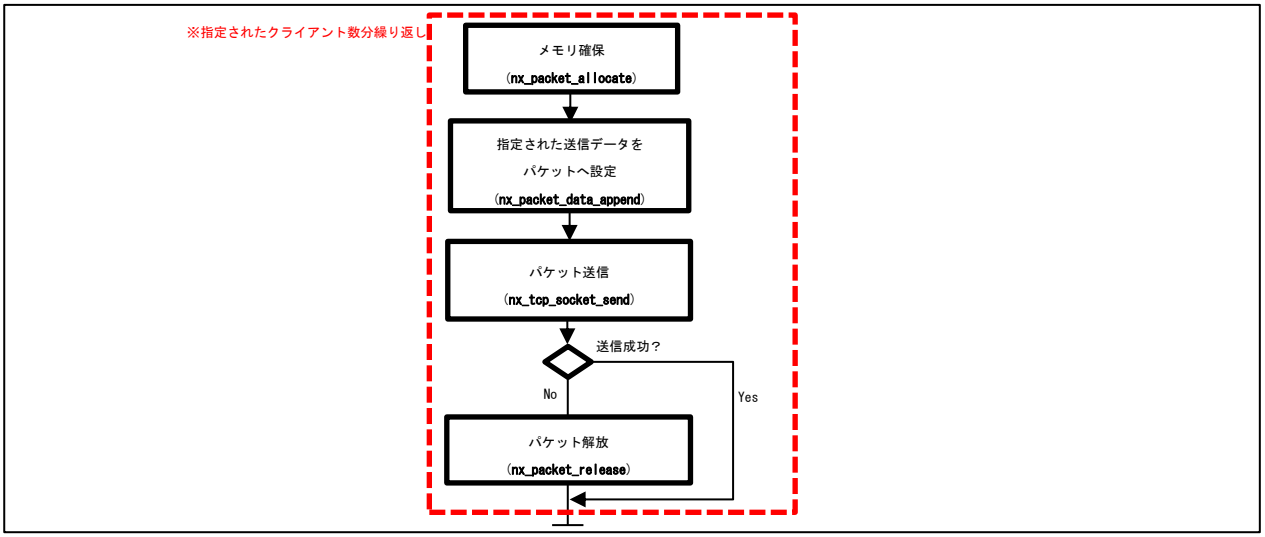

図 13　パケット送信時のシーケンス

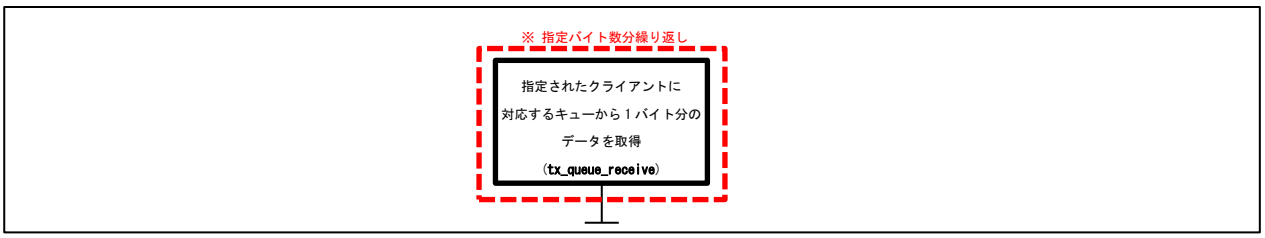

図 14　受信パケット取得時のシーケンス

4.4.4　通信終了時のシーケンス

マルチクライアントサーバとの通信を終了するためのシーケンスは図 15 の通りです。また、対応する関数は、client/src/client/client.c 内の client_close()です。

シーケンスの概要を以下に示します。

① nx_tcp_socket_disconnect()をコールし、切断要求を送信する。

② nx_tcp_client_unbind()をコールし、バインドを解除する。

③ nx_tcp_socket_delete()をコールし、ソケットを削除する。

④ tx_queue_delete()をコールし、受信パケット用のキューを削除する。

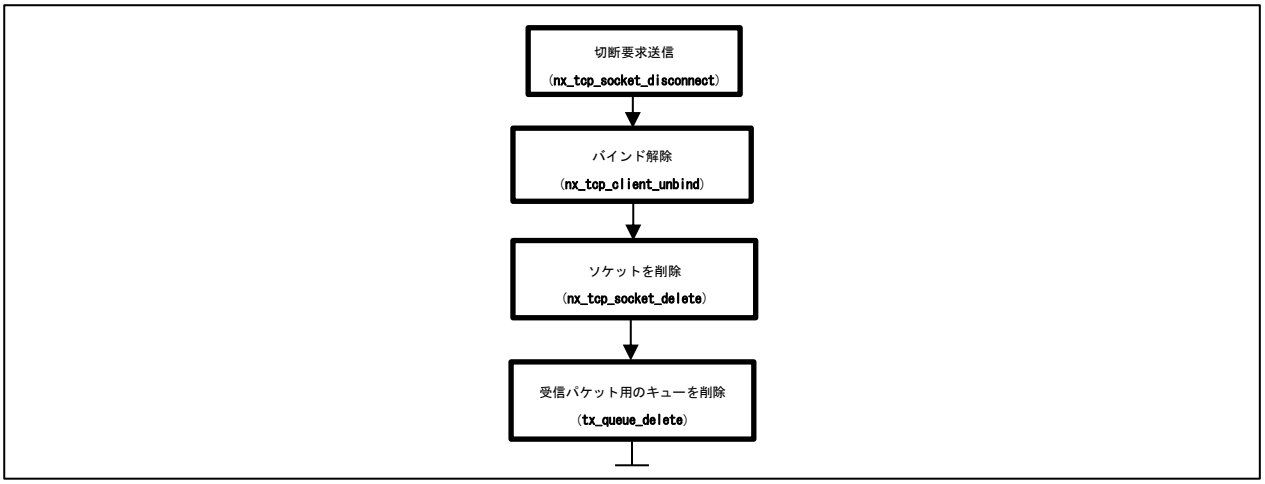

図 15　通信終了時のシーケンス

4.4.5　アプリのシーケンス

アプリは、先述の関数（client_open(), client_send(), client_receive(), client_close()）を組み合わせることで、2 章で説明したクライアントの動作を実現しています。

まず初期化時の動作は図 16 の通りです。App Thread は最初に client_open()をコールし、初期化及びマルチクライアントサーバへ接続します。その後、マルチクライアントサーバからクライアント名が要求されるため、自身のクライアント名を送信します。その後 LED1 を点灯します。マルチクライアントサーバからのパケット受信を行うための Sub Thread を生成します。

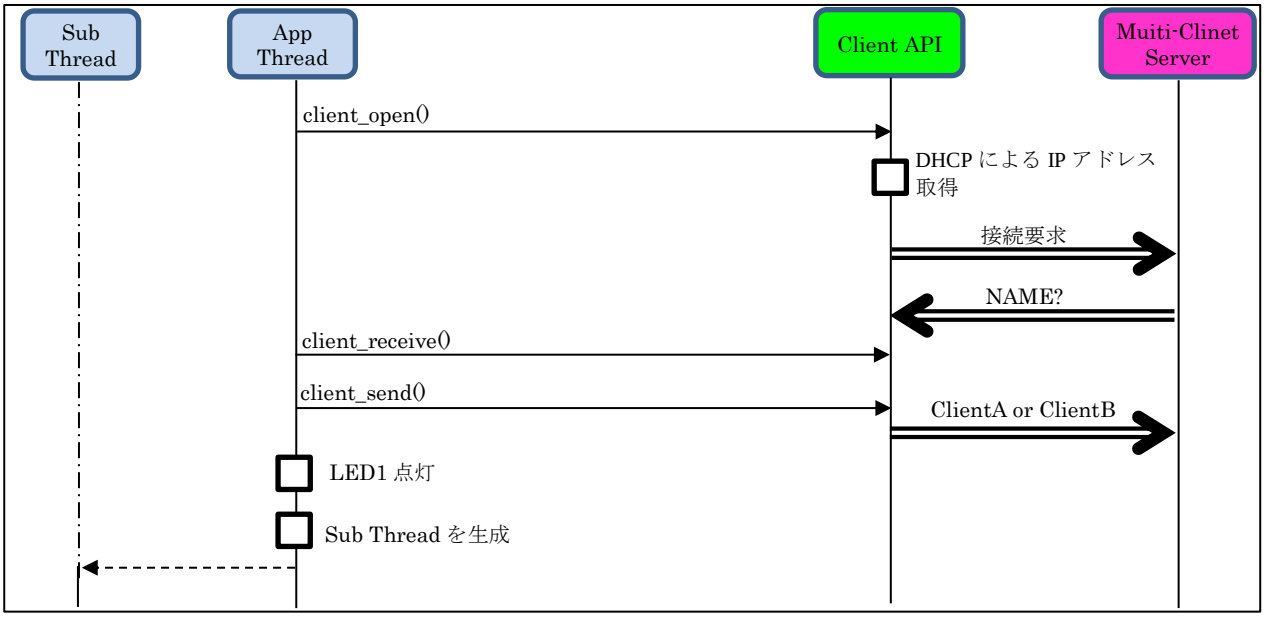

図 16　初期化時のシーケンス

初期化後の App Thread の動作は以下の通りです。S4 が押される度に client_send()をコールすることで、マルチクライアントサーバに対して LED トグルコマンドを送信します。また、S4 を押した回数が 5 回を超えた場合もしくは、マルチクライアントサーバとの通信が終了した場合、client_close()をコールして、マルチクライアントサーバに対して切断要求を送信します（既に通信が終了している場合 client_close()をコールしても切断要求は送信せず、client_open()時に生成したリソースの解放のみ実施）。そして最後に、LED1 を消灯します。

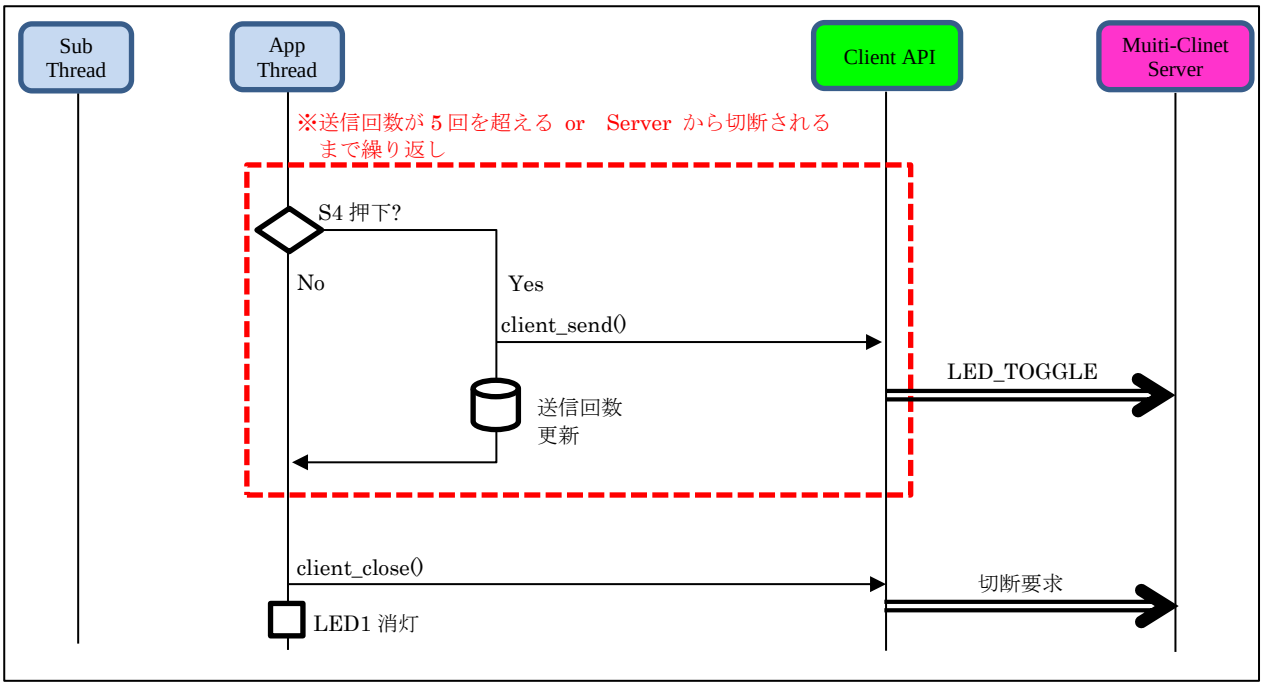

図 17　App Thread の動作

また、初期化時に生成した Sub Thread の動作は図 18 の通りです。Sub Thread は client_receive()をコールして、マルチクライアントサーバからの LED トグルコマンドを受信します。受信後は LED3 をトグルします。また、対象のマルチクライアントサーバから切断要求があった場合は、Sub Thread を終了します。

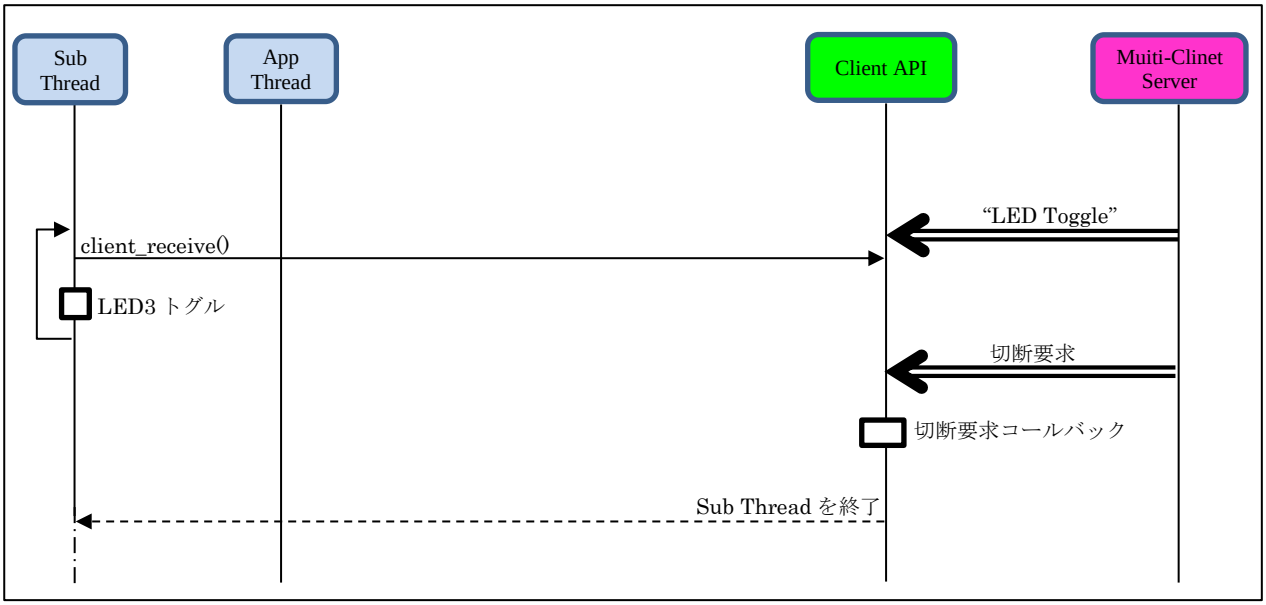

図 18　Sub Thread の動作

4.5　MAC アドレス変更

4.5.1　Synergy Configuration が提供する設定方法

Synergy Configuration では、NetX Port ETHER on sf_el_nx のプロパティにて、以下の 2 通りの Ethernet の MAC アドレスを変更する方法を提供しています（図 19）。

A)　ビルド時に MAC アドレスを設定する方法
⇒　「Channel 0/1 MAC Address High/Low Bits」プロパティの値を変更することで可能です。

B)　初期化時にユーザアプリから MAC アドレスを変更する方法
⇒　「Callback」プロパティにコールバック関数を設定することで可能です。このコールバック関数は電源投入後の初期化時に呼び出されます。

また、本書付属のサンプルプログラムでは B)の方法を用いています。

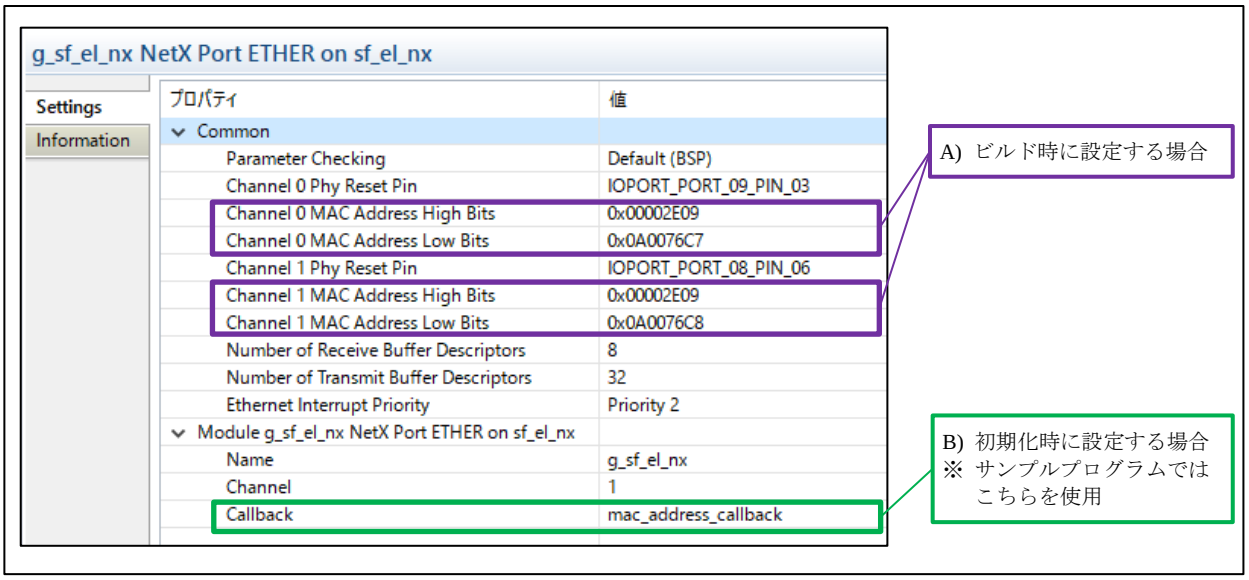

図 19　SSP での MAC アドレス設定方法

4.5.2　サンプルプログラムでの設定方法

本書付属のサンプルプログラムでは、NetX Port ETHER on sf_el_nx の「Callback」プロパティに指定したコールバック内で、データフラッシュに格納されている値を読み取り MAC アドレスを設定しています。対応する関数は、{multi_client_server | client}/src/app_thread_entry.c 内の mac_address_callback()です。

サンプルプログラムの MAC アドレスの構成及び、マルチクライアントサーバ・クライアント A／B に割り当てている MAC アドレスは表 4,5 の通りです。本書のサンプルプログラムは LAN 内で通信を行うため、MAC アドレスはローカルアドレスを割り当てていますが、ユニバーサルアドレスを想定した構成としています。また、インターネット上の機器と直接通信を行う機器の場合は、ユニバーサルアドレスとし OUI(Organizationally Unique Identifier)を IEEE より取得する必要があります。詳細は、以下の URL を参照してください。

- http://ieee802.org/secmail/pdfYD89wBpRqH.pdf

表 4　MAC アドレス構成

第 1 オクテット	第 2 オクテット	第 3 オクテット	第 4 オクテット	第 5 オクテット	第 6 オクテット
ベンダ ID (OUI)			機種 ID	シリアル ID	

表 5　各端末に割り当てている MAC アドレス

端末	第 1 オクテット	第 2 オクテット	第 3 オクテット	第 4 オクテット	第 5 オクテット	第 6 オクテット
マルチクライアントサーバ	2E:09:0A			00	00:00	
クライアント A				01	00:00	
クライアント B					00:01	

また本書では、データフラッシュに格納する情報はシリアル ID のみを格納しています（表 6）。

表 6　データフラッシュ内の構成

アドレス	値
0x40100000	シリアル ID　下位 1byte
0x40100001	シリアル ID　上位 1byte

4.5.3　Renesas Flash Programmer によるデータフラッシュへの書き込み手順

本書では、4.5.2 で説明したシリアル ID を Renesas Flash Programmer（以降、RFP）によってデータフラッシュへ書き込む手順について説明します。RFP については、参考文献[6]を参照してください。
書き込み手順は以下の通りです。マルチクライアントサーバ、クライアント A・B それぞれの SK-S7G2 ボードに対して実施してください。

1. SK-S7G2 ボードの J1 を 2-3 ショートに設定します。
2. J5 とホスト PC を USB ケーブルで接続します。
3. J19 にケーブルを接続し、ボードに電源を投入します。
4. ホスト PC で RFP を起動します。
5. 「ファイル」－「新しいプロジェクトの作成」を選択し、以下を設定します。
   ・マイクロコントローラ：Synergy
   ・プロジェクト名：任意
   ・ツール詳細：適切な COM
6. 「接続」ボタンを押します。

   以下のエラー・メッセージが表示された場合は、SK-S7G2 をリセットした後に、再度、「接続」ボタンを押します。
   ・エラー(E3000105): デバイスから応答がありません。
7. 「ブロック設定」タブを選択し、Data Flash 1 内の Block 134 以外の Erase, P.V のチェックを外します。
   ※既にサンプルプログラムが書き込まれている場合、Code Flash 1 の消去を避けるためです。
8. 「操作タブ」の「プログラムファイル」にマルチクライアントサーバ・クライアントに対応する以下の.mot ファイルを指定します。

   ・マルチクライアントサーバ：multi_client_server/Serial_ID_server.mot
   ・クライアント A：client/Serial_ID_clientA.mot
   ・クライアント B：client/Serial_ID_clientB.mot
   ※これらの.mot ファイルに各シリアル ID の値と書き込み先のアドレスが指定されています。
9. SK-S7G2 をリセットした後、[スタート]ボタンを押します。
10. 書き込み完了後、RFP を終了し、SK-S7G2 ボードの J1 を 1-2 ショートに戻します。

5. Appendix: マルチクライアントサーバに 3 つ以上クライアントを接続するためには

本書付属のサンプルプログラム（multi_client_server.zip）では、2 つのクライアントと通信していますが、3 つ以上のクライアントと通信する場合の変更点及び注意点を説明します。

マルチクライアントサーバが許容する最大クライアント数は 8 となっています。これは、クライアント情報を 8 ビットの変数で管理し、各ビットがクライアントに対応しているためです。クライアント情報は、multi_client_server/src/multi_client_server/multi_client_server.c 内の 48 行目 uint8_t current_clients で管理しているため、8 つ以上のクライアントと通信したい場合、この変数の型を uint8_t より大きい型へ修正してください。また、ソケットやキュー、そして接続要求・切断要求受信時に使用するスレッドのスタックなどはクライアント数分、配列として管理しています。その要素数は multi_client_server.c 内の 32 行目 MAX_CLIENTS で定義しているため、クライアント数に応じて変更する必要があります。

また、注意点としては最大クライアント数を大きくしすぎると、上記のようにクライアント数分リソース（特にスレッドのスタック）を必要とするため、RAM 容量をひっ迫してしまうことです。そのため、ユーザアプリの RAM 使用量も考慮し、マルチクライアントサーバ内部で確保しているリソース（特にスレッドのスタック）のサイズを適宜調整する必要があります。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2017-12-28	-	初版

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子
   （または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、その他の不適切に使用しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することはできません。たとえ、意図しない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル（無人航空機を含みます。）の開発、設計、製造、使用もしくは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、かつ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。
   当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様の転売、貸与等により、本書（本ご注意書きを含みます。）記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は一切その責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

(Rev.3.0-1 2016.11)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/