【注意事項】
RX ファミリ 組み込み用 TCP/IP M3S-T4-Tiny

R20TS0227JJ0100
Rev.1.00
2017.11.16 号

概要

RX ファミリ 組み込み用 TCP/IP M3S-T4-Tiny (注) の使用上の注意事項を連絡します。

1. Ping Reply パケットに関する注意事項
2. LAN ネットワーク環境に関する注意事項
3. select()関数のタイムアウト設定値に関する注意事項
4. sendto()関数で送信する UDP パケットの宛先 IP アドレスに関する注意事項

注：詳細は各項の「該当製品」をご参照ください。

1. Ping Reply パケットに関する注意事項

1.1 該当製品

- RX ファミリ 組み込み用 TCP/IP M3S-T4-Tiny を用いたサンプルプログラム Firmware Integration Technology V.1.00～V.1.06
- RX ファミリ 組み込み用 TCP/IP M3S-T4-Tiny Firmware Integration Technology V.1.03～V.2.06

1.2 該当 MCU

RX ファミリ

1.3 内容

M3S-T4-Tiny の Ping Reply パケットの送信処理に誤りがあり、IP ヘッダの IP Total length フィールドに間違った値を設定します。そのため、Ping Request を送信した端末が Ping Reply を異常なパケットと認識し、正常な応答とみなさない場合があります。

1.4 発生条件

受信した Ping Request パケットに含まれる IP ヘッダの IP Total length が 45 以下の場合に発生します。

1.5 回避策

回避策はありません。

1.6 恒久対策

今後のバージョンで改修予定です。

2. LAN ネットワーク環境に関する注意事項

2.1 該当製品

- RX ファミリ 組み込み用 TCP/IP M3S-T4-Tiny を用いたサンプルプログラム Firmware Integration Technology V.1.00～V.1.06
- RX ファミリ Ethernet ドライバと組み込み用 TCP/IP M3S-T4-Tiny のインタフェース変換モジュール Firmware Integration Technology V.1.00～V.1.06

2.2 該当 MCU

RX ファミリ

2.3 内容

10BASE 半二重モードで通信する LAN ネットワーク環境でサンプルプログラムを動作させた場合、ボードによっては送信したパケットが RX マイコンに返ります。そのため、DHCP 機能が正常に動作しません。

2.4 発生条件

10BASE 半二重モードで接続した場合に発生します。

2.5 回避策

受信したパケットの送信 MAC アドレスが、RX マイコンに設定した MAC アドレスと同じであれば、受信したパケットを破棄します。

以下の関数に赤字部分を追加してください。

■ r_t4_driver_rx¥src¥t4_driver.c: lan_read()

```
H lan_read(UB lan_port_no, B **buf)
{
    int32_t driver_ret;
    H return_code;
    UB *data;

    driver_ret = R_ETHER_Read_ZC2(lan_port_no, (void **)buf);
    if (driver_ret > 0)
    {
        data = (B *)*buf;
        if(0 == memcmp(&data[6],&_myethaddr[lan_port_no],6))
        {
            rcv_buff_release(lan_port_no);
            return_code = -1;
            return return_code;
        }
        t4_stat[lan_port_no].t4_rec_cnt++;
        t4_stat[lan_port_no].t4_rec_byte += (UW)driver_ret;
        return_code = (H)driver_ret;
    }
    else if (driver_ret == 0)
```

2.6 恒久対策

今後のバージョンで改修予定です。

3. select()関数のタイムアウト設定値に関する注意事項

3.1 該当製品

- RX ファミリ 組み込み用 TCP/IP M3S-T4-Tiny を用いたサンプルプログラム Firmware Integration Technology V.1.05
- RX ファミリ 組み込み用 TCP/IP M3S-T4-Tiny ソケット API モジュール Firmware Integration Technology V.1.31

3.2 該当 MCU

RX ファミリ

3.3 内容

select()関数の第 5 引数に指定するタイムアウト時間の処理に誤りがあり、以下 2 点の注意事項があります。

(1) 指定した時間より短い時間でタイムアウトが発生します。

(2) select()関数が終了するまで最大 10 ミリ秒かかります。

3.4 発生条件

(1) 3.3 (1)の発生条件

常に発生します。

以下に発生例を記します。

```
struct timeval select_timer;
/* 1.5 second */
select_timer.tv_sec = 1;
select_timer.tv_usec = 500000;
nready = select(1, NULL, NULL, NULL, &select_timer);
```

上記設定の場合においても、tv_usec メンバの設定値が無視されて 0 を設定した場合と同じ動作になります。そのため、指定した時間より短い時間でタイムアウトが発生します。

(2) 3.3 (2)の発生条件

tv_sec および tv_usec メンバにともに 0 を設定した場合に発生します。

3.5 回避策

以下の関数に赤字部分を追加してください。

■ソケット API モジュールのソースコード”r_socket.c”の select()関数

```
  uint32_t polling = 0;

  (中略)
  if ( timeout == NULL )
  {
  }
  else if ((timeout->tv_usec >= 1000000) || (timeout->tv_usec < 0) ||
(timeout->tv_sec < 0))
  {
     timeout = NULL;
  }
  else if ((timeout->tv_sec == 0) && (timeout->tv_usec == 0))
  {
     polling = 1;
  }
  else
  {
  //       timeout->tv_usec /= 10000;
  }
  timer1 = tcpudp_get_time();

  (中略)

        if (tot_count > 0)
        {
           break;
        }
        if(polling == 1)
        {
           break;
        }
        if (timeout != NULL)
        {
```

3.6 恒久対策

今後のバージョンで改修予定です。

4. sendto()関数で送信する UDP パケットの宛先 IP アドレスに関する注意事項

4.1 該当製品

- RX ファミリ 組み込み用 TCP/IP M3S-T4-Tiny を用いたサンプルプログラム Firmware Integration Technology V.1.05
- RX ファミリ 組み込み用 TCP/IP M3S-T4-Tiny ソケット API モジュール Firmware Integration Technology V.1.31

4.2 該当 MCU

RX ファミリ

4.3 内容

UDP パケットを送信する sendto()関数で指定された宛先 IP アドレスの処理に誤りがあり、以下の注意事項があります。

(1) 間違った IP アドレスに UDP パケットを送信する場合があります。

4.4 発生条件

ソケットが sendto()関数を実行するタイミングと同時に、別の IP アドレスから UDP パケットを受信した場合に発生します。

4.5 回避策

送信制御用と受信制御用に IP アドレス情報を分離して制御します。

以下の赤字部分を修正してください。

(1) r_socket_rx_if.h

修正前

```
typedef struct _tag_BSDSocket
{
    BSD_STATE   state;      /* BSD socket states */
    uint32_t    T4status;    /* T4 current status */
    int         event;      /* T4 event occuring in callback */
    ID          socket_type; /* SOCK_STREAM or SOCK_DGRAM */
    ID          backlog;    /* No Used */
    T_IPV4EP    dstaddr;    /* Partners Address */
    TMO         tmout;      /* Time out */
    (以下省略)
} BSDSocket;
```

修正後

```
typedef struct _tag_BSDSocket
{
    BSD_STATE   state;      /* BSD socket states */
    uint32_t    T4status;    /* T4 current status */
    int         event;      /* T4 event occuring in callback */
    ID          socket_type; /* SOCK_STREAM or SOCK_DGRAM */
    ID          backlog;    /* No Used */
    T_IPV4EP    dstaddr;    /* Partners Address */
    T_IPV4EP    udpsndaddr; /* UDP Send Address */
    TMO         tmout;      /* Time out */
    (以下省略)
} BSDSocket;
```

(2) r_socket_rx.c　R_SOCKET_Open()

修正前

```
        sockets[i].dstaddr.ipaddr = 0;
        sockets[i].dstaddr.portno = 0;
        sockets[i].tmout = 0;  /* default is TMO_POL */
        sockets[i].T4proc = 0;
        sockets[i].rcvLen = 0;
```

修正後

```
        sockets[i].dstaddr.ipaddr = 0;
        sockets[i].dstaddr.portno = 0;
        sockets[i].udpsndaddr.ipaddr = 0;
        sockets[i].udpsndaddr.portno = 0;
        sockets[i].tmout = 0;  /* default is TMO_POL */
        sockets[i].T4proc = 0;
        sockets[i].rcvLen = 0;
```

(3) r_socket_rx.c　connect()

修正前

```
    else
    {
        /* UDP: remote port is used as a filter only.  no need to call
connect() */
        /* TODO any check for addr range, port range? */
        sockets[sock].dstaddr.portno = remote_port;
        sockets[sock].dstaddr.ipaddr = remote_ip;
        sockets[sock].state = BSD_CONNECTED;
    }
    return E_OK;
```

修正後

```
    else
    {
        /* UDP: remote port is used as a filter only.  no need to call
connect() */
        /* TODO any check for addr range, port range? */
        sockets[sock].dstaddr.portno = remote_port;
        sockets[sock].dstaddr.ipaddr = remote_ip;
        sockets[sock].udpsndaddr.portno = remote_port;
        sockets[sock].udpsndaddr.ipaddr = remote_ip;
        sockets[sock].state = BSD_CONNECTED;
    }
    return E_OK;
```

(4) r_socket_rx.c sendto()

修正前

```
        if ( sockets[sock].state < BSD_CONNECTED )
        {
            sockets[sock].dstaddr.portno = addr->sin_port;
            sockets[sock].dstaddr.ipaddr = addr->sin_addr.S_un.S_addr;
        }

        memcpy( sockets[sock].snd_buf, buffer, length );
        sockets[sock].sndLen = 0;
        sockets[sock].sndSz = length;
        sockets[sock].pending_sndlen = length;
        sockets[sock].T4proc &= ~(T4_PROC_SND_END);
        sockets[sock].T4proc |= (T4_PROC_SND_START);
        ercd = udp_snd_dat( cepid, (T_IPV4EP*) & sockets[sock].dstaddr,
(VP)sockets[sock].snd_buf, length, TMO_NBLK );
        if ( ercd == E_WBLK )
        {
            ercd = length;  /* Return length of data even though it may
not been sent*/

        }
```

修正後

```
        if ( sockets[sock].state < BSD_CONNECTED )
        {
            sockets[sock].udpsndaddr.portno = addr->sin_port;
            sockets[sock].udpsndaddr.ipaddr = addr-
>sin_addr.S_un.S_addr;
        }

        memcpy( sockets[sock].snd_buf, buffer, length );
        sockets[sock].sndLen = 0;
        sockets[sock].sndSz = length;
        sockets[sock].pending_sndlen = length;
        sockets[sock].T4proc &= ~(T4_PROC_SND_END);
        sockets[sock].T4proc |= (T4_PROC_SND_START);
        ercd = udp_snd_dat( cepid, (T_IPV4EP*) &
sockets[sock].udpsndaddr, (VP)sockets[sock].snd_buf, length,
TMO_NBLK );
        if ( ercd == E_WBLK )
        {
            ercd = length;  /* Return length of data even though it may
not been sent*/

        }
```

(5) r_socket_rx.c　t4_udp_generic_callback()

修正前

```
            case TFN_UDP_SND_DAT:
                sockets[i].T4proc &= ~((uint32_t)T4_PROC_SND_START);
                sockets[i].event = SOCKET_SND_DAT;
                sockets[i].sndLen += ercd;
                remain_size = sockets[i].sndSz - sockets[i].sndLen;
                if ( remain_size > 0 )
                {
                    udp_snd_dat(cepid,  &sockets[i].dstaddr,
(VP)(&sockets[i].snd_buf[sockets[i].sndLen]),
                                remain_size, TMO_NBLK);
                    sockets[i].T4proc |= ((uint32_t)T4_PROC_SND_START);
                }
```

修正後

```
            case TFN_UDP_SND_DAT:
                sockets[i].T4proc &= ~((uint32_t)T4_PROC_SND_START);
                sockets[i].event = SOCKET_SND_DAT;
                sockets[i].sndLen += ercd;
                remain_size = sockets[i].sndSz - sockets[i].sndLen;
                if ( remain_size > 0 )
                {
                    udp_snd_dat(cepid,  &sockets[i].udpsndaddr,
(VP)(&sockets[i].snd_buf[sockets[i].sndLen]),
                                remain_size, TMO_NBLK);
                    sockets[i].T4proc |= ((uint32_t)T4_PROC_SND_START);
                }
```

(6) r_socket_rx.c　reset_socket ()

修正前

```
    if ( sock != -1 )
    {
        sockets[sock].state = BSD_CLOSED;
        sockets[sock].T4status = T4_CLOSED;
        sockets[sock].event = -1;
        sockets[sock].socket_type = 0;
        sockets[sock].backlog = 0;
        sockets[sock].dstaddr.ipaddr = 0;
        sockets[sock].dstaddr.portno = 0;
        sockets[sock].tmout = 0;  //default is TMO_POL
```

修正後

```
    if ( sock != -1 )
    {
        sockets[sock].state = BSD_CLOSED;
        sockets[sock].T4status = T4_CLOSED;
        sockets[sock].event = -1;
        sockets[sock].socket_type = 0;
        sockets[sock].backlog = 0;
        sockets[sock].dstaddr.ipaddr = 0;
        sockets[sock].dstaddr.portno = 0;
        sockets[sock].udpsndaddr.ipaddr = 0;
        sockets[sock].udpsndaddr.portno = 0;
        sockets[sock].tmout = 0;  //default is TMO_POL
```

4.6　恒久対策

今後のバージョンで改修予定です。

以上

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2017.11.16	-	新規発行

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）

■総合お問い合わせ先
https://www.renesas.com/contact/

本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
過去のニュース内容は発行当時の情報をもとにしており、現時点では変更された情報や無効な情報が含まれている場合があります。
ニュース本文中の URL を予告なしに変更または中止することがありますので、あらかじめご承知ください。

すべての商標および登録商標は、それぞれの所有者に帰属します。

© 2017 Renesas Electronics Corporation. All rights reserved.
TS Colophon 2.0