

Contents

- [1.0 Introduction](#)
- [2.0 User-Supplied Functions](#)
- [3.0 Code Description](#)
- [4.0 Code](#)

1.0 Introduction

This application note provides sample test code for MT90224 16 port PHY IMA /TC device. It gives typical sequence to initialize the device, and to create basic IMA configuration. The main purpose of this document is to allow users to verify the hardware design before porting IMA - Core software. It is important to note that the code provided here is not enough to put the device into full operation. It should only be used for test purposes. Full IMA - Core software package together with MT90224 is needed in order to incorporate Zarlink IMA solution in user's environment.

The application note should be used together with MT90224/3/2 data sheet. Detailed description of each register can be found in the data sheet.

The sample code is also applicable to MT90223, MT90222 and ZL30225/6/7 with minor changes.

For programming of MT90224/3/2 in TC mode, users should refer to application note TN90225.1

2.0 User-Supplied Functions

It is assumed that the user should supply low-level access functions "write_reg", "read_reg" and "Sleep".

The function "write_reg" writes data to specified register.

Function: void write_reg (U16 Register, U16 Data)

Inputs: Register - register offset.

Data - value to be written

Returns: None

Comments: U16 is unsigned 16-bit integer

The function "read_reg" reads data from specified register.

Function: U16 read_reg (U16 Register)

Inputs: Register - register offset.

Returns: Read data.

Comments: U16 is unsigned 16-bit integer

The function "Sleep" pauses execution for specified number of milliseconds.

Function: void Sleep (int time)

Inputs: time - number of milliseconds

Returns: None

The following constants need to be defined as well.

```
/* define constants */
#define NUM_LINKS      16 /* maximum number of links
                           MT90224 - 16
                           MT90223 - 8
                           MT90222 - 4 */
#define NUM_GROUPS     8  /* maximum number of groups
                           MT90224 - 8
                           MT90223 - 8
                           MT90222 - 4 */
```

Any line marked as "### user changeable ###" indicates that user can write other values to the register based on his hardware configuration or feature selection.

3.0 Code Description

All necessary steps are implemented in the function Setup8g2l. It performs next steps:

- Resets the device.
- Initializes general registers
- Enables digital loopback and initialize TDM registers.
- Configures 8 IMA groups with 2 links each.
- Configures and enables the UTOPIA interface.

The IMA groups are configured as follows:

Group 0 - links 0,1. The reference link is link 0.

Group 1 - links 2,3. The reference link is link 2.

.....

Group 7 links 14,15. The reference link is link 14.

After running the function, the user will be ready to send and receive UTOPIA traffic to/from MT90224 and to verify the operation of the hardware.

4.0 Code

```
void Setup8g2l(void)
{
    U16 temp;                // U16 - unsigned 16 bit integer
    S16 flag;                // S16 - signed 16 bit integer
    U16 Base,link, group, i;

    // reset
    temp = 0x00E0;
    write_reg( 0x0299, temp );

    Sleep(1);                //1 ms wait.

    // Initialize for regular operation
    temp = 0x0100;
    write_reg( 0x0333, temp );

    // SRAM Control register      ### user changeable ###
    temp = 0x0005;              // 2x512K SRAM Size
    write_reg( 0x0299, temp );

    // wait for cell RAM ready
    do
    {
        Sleep(1);              // 1 ms
        temp = read_reg(0x0080);
    } while ( (temp & 0x0080) == 0 );

    // Utopia Output User Defined Byte register
    temp = 0x0055;      ### user changeable ###
    write_reg( 0x0012, temp );

    // Disable all Utopia output link PHYs
    temp = 0x0000;
    write_reg( 0x0010, temp );

    // Disable all Utopia output group PHYs
    // set 16-bit mode
    temp = 0x0000;
    write_reg( 0x0011, temp );

    // Disable all Utopia input link PHYs
    temp = 0x0000;
    write_reg( 0x0050, temp );

    // Disable all Utopia input group PHYs
    temp = 0x0000;
    write_reg( 0x0051, temp );
```

```
// Disable debugging mode
temp = 0x0000;
write_reg( 0x0108, temp );

// Rx Auto sync reg - needed only if working without frame pulse
temp = 0x0036;          ### user changeable ###
write_reg( 0x0741, temp );

// LCD register
temp = 0x000C;
write_reg( 0x00C8, temp );

// configure Tx Link FIFO size (for UNI mode set to non zero)
for (link = 0; link < NUM_LINKS; link++)
    if (link < 8){
        temp = 0x0000|( (read_reg( 0x008B +link) & 0xFFF0));
        write_reg( 0x008B +link, temp);
    }
    else{
        temp = 0x0000|( (read_reg( 0x008B +link- 0x8) & 0xF0FF));
        write_reg( 0x008B +link- 0x8, temp);
    }

for (group = 0; group < NUM_GROUPS; group++) {

    // Setup guardband
    temp = 0x000A;          // ### user changeable ###
    write_reg( 0x0287 + group, temp );

    // Setup max operational delay LSB value (MSB is 0)
    temp = 0x005B;          // ### user changeable ###
    write_reg( 0x029A + group, temp );

    // Configure Utopia TX FIFO size for IMA mode
    if (group < 4){
        temp = 0x0002|( (read_reg( 0x0093 + group) & 0xFFF0));
        write_reg( 0x0093 + group, temp);
    }
    else{
        temp = 0x0200|( (read_reg( 0x0093 + group- 0x4) & 0xF0FF));
        write_reg( 0x0093 + group - 0x4, temp);
    }

    //Setup TX and RX integration period, 0x000C for E1;
    // ### user changeable ###
    if (group < 4){
        temp = 0x000C|( (read_reg( 0x0403 + group) & 0xFFF0));
        write_reg( 0x0403 + group, temp);
    }
}
```

```

        temp = 0x000C | ( (read_reg( 0x0219 + group) & 0xFFF0));
        write_reg( 0x0219 + group, temp);
    }
    else{
        temp = 0x0C00 | ( (read_reg( 0x0403 + group- 0x4) & 0xF0FF));
        write_reg( 0x0403 + group - 0x4, temp);
        temp = 0x0C00 | ( (read_reg( 0x0219 + group- 0x4) & 0xF0FF));
        write_reg( 0x0219 + group - 0x4, temp);
    }
} // end for - setup for group

/*-----*
 * Tx setup
 *-----*/

// TDM Tx mapping registers
temp = 0xFFFE; // ### user changeable ###
for(link = 0; link < NUM_LINKS; link++) {
    write_reg( 0x0610 + link, temp );
    write_reg( 0x0620 + link, temp );
}

// Link ID
for (link = 0; link < NUM_LINKS; link++){
    if (link < 8){
        temp = (0x0000 + link) | ( (read_reg( 0x0336 + link) & 0xFFE0));
        write_reg( 0x0336 + link, temp);
    }
    else{
        temp = ((0x0 + link) << 8) | ((read_reg(0x0336+link-0x8) & 0xE0FF));
        write_reg( 0x0336 + link - 0x8, temp);
    }
}

// ICP cell offset - test purposes only.
for (link = 0; link < NUM_LINKS; link++){
    if (link < 8){
        temp = (0x0000 + link) | ((read_reg( 0x0310 + link) & 0xFF00));
        write_reg( 0x0310 + link, temp);
    }
    else{
        temp = ((0x0+link) << 8) | ((read_reg( 0x0310 + link-0x8) & 0x00FF));
        write_reg( 0x0310 + link - 0x8, temp);
    }
}

// TX TDM Link Control register
temp = 0x48A4; // ### user changeable ###
for(link = 0; link < NUM_LINKS; link++) {

```

```
    write_reg( 0x0600 + link, temp );
}

for (group = 0 ; group < NUM_GROUPS; group ++ ) {

    // Tx Group Control Mode Reg
    // CTC, group reference link is chosen to be equal to group number*2
    temp = 0x00B0;
    write_reg( 0x0300 + group, temp|(group*2) );

    // Tx IMA Control register - TDM FIFO size for links in IMA mode
    if (group < 4){
        temp = 0x0031|( (read_reg( 0x0321 + group) & 0xFF00));
        write_reg( 0x0321 + group, temp);
    }
    else{
        temp = 0x3100|( (read_reg( 0x0321 + group- 0x4) & 0x00FF));
        write_reg( 0x0321 + group - 0x4, temp);
    }

    for ( i = 0; i < 2; i++){
        link = i + group * 2;

        // TX Link Control
        // Set IMA group, don't enable IMA mode,
        if (link < 8){
            temp = 0x0008|( (read_reg( 0x0318 + link) & 0xFFF0));
            write_reg( 0x0318 +link, temp|group);
        }
        else{
            temp = (0x0800|(group<<8))|((read_reg(0x0318 +link-8)& 0xF0FF));
            write_reg( 0x0318 + link- 8, temp);
        }
    }

    // Write a Tx ICP cell for each group
    Base = 0x0500 + group * 0x20;
    temp = 0x0000;        // cell header 1 & cell header 2
    write_reg( Base + 0x0000, temp);
    temp = 0x0B00;        // cell header 3 & cell header 4
    write_reg( Base + 0x0001, temp);
    temp = 0x0100;        // cell header 5 & OAM label
    write_reg( Base + 0x0002, temp);
    temp = 0x0000;        // cell link id & IMA Frame Sequence number
    write_reg( Base + 0x0003, temp);
    temp = 0x0000;        // icp cell offset & link stuff indication
    write_reg( Base + 0x0004, temp);
    temp = 0x0000;        // status control change indication & ima id
    write_reg( Base + 0x0005, temp);
}
```

```
temp = 0x0000;        // group status control & transmit timing
write_reg( Base + 0x0006, temp);
temp = 0xAA00;        // tx test control & tx test pattern
write_reg( Base + 0x0007, temp);
temp = 0x5511;        // rx test pattern & link 0 info
write_reg( Base + 0x0008, temp);
temp = 0x1111;        // link_1_info & link 2 info
write_reg( Base + 0x0009, temp);
temp = 0x1111;        // link_3_info & link 4 info
write_reg( Base + 0x000a, temp);
temp = 0x1111;        // link_5_info & link 6 info
write_reg( Base + 0x000b, temp);
temp = 0x1111;        // link_7_info & link 8 info
write_reg( Base + 0x000c, temp);
temp = 0x1111;        // link_9_info & link 10 info
write_reg( Base + 0x000d, temp);
temp = 0x1111;        // link_11_info & link 12 info
write_reg( Base + 0x000e, temp);
temp = 0x1111;        // link_13_info & link 14 info
write_reg( Base + 0x000f, temp);
temp = 0x1111;        // link_15_info & link 16 info
write_reg( Base + 0x0010, temp);
temp = 0x1111;        // link_17_info & link 18 info
write_reg( Base + 0x0011, temp);
temp = 0x1111;        // link_19_info & link 20 info
write_reg( Base + 0x0012, temp);
temp = 0x1111;        // link_21_info & link 22 info
write_reg( Base + 0x0013, temp);
temp = 0x1111;        // link_23_info & link 24 info
write_reg( Base + 0x0014, temp);
temp = 0x1111;        // link_25_info & link 26 info
write_reg( Base + 0x0015, temp);
temp = 0x1111;        // link_27_info & link 28 info
write_reg( Base + 0x0016, temp);
temp = 0x1111;        // link_29_info & link 30 info
write_reg( Base + 0x0017, temp);
temp = 0x6A11;        // link_31_info & unused
write_reg( Base + 0x0018, temp);
temp = 0x1111;        // end to end channel & crc_error 1
write_reg( Base + 0x0019, temp);
temp = 0x0011;        //crc_error_2 & not used
write_reg( Base + 0x001a, temp);

// start ICP cell transfer
temp = 0x00FF & ~(1 << group);
write_reg( 0x0086, temp);
Sleep (36 * 3);        // 36*3 ms

// Tx Links Activation
```

```

for (i = 0; i < 2; i++){

link = i + group*2;

// Tx Add Link Control register
if (link != group * 2){
    temp = 0x0108 + (0x0000|group);
    write_reg( 0x333, temp );
}

// Enable IMA mode - Tx Link Control Register
if (link < 8){
    temp = 0x0000|( (read_reg( 0x0318 +link) & 0xFFF7));
    write_reg( 0x0318 +link, temp);
}
else{
    temp = 0x0000|( (read_reg( 0x0318 +link-8) & 0xF7FF));
    write_reg( 0x0318 +link -8, temp);
}

// Wait until status bit is set as IMA mode
flag = 0;
while (flag < 100)
{
    // Tx IMA Status
    temp = read_reg( 0x0346);
    temp = (temp >> link) & 0x1;
    if (temp == 0) break;
    flag ++;
    Sleep (2); // 2 ms
}
if (flag == 100) printf("\nTx IMA mode error for link%x\n", link);

//Tx Add Link Control - non reference link only
if (link != group*2){
    temp = 0x0100|( (read_reg( 0x0333) & 0xFEF7));
    write_reg( 0x0333, temp);
}
} // end for_loop tx link activation.
} // end group for_loop.

/*-----*
* RX setup
*-----*/

// TDM Rx mapping registers
temp = 0xFFFE; // ### user changeable ###
for(link = 0; link< NUM_LINKS; link++) {
    write_reg( 0x0710 +link, temp );
}

```



```
    write_reg( 0x0720 +link, temp );
}

// Rx PCM port configure, TDM digital loopback enabled
temp = 0x01A4; // ### user changeable ###
for(link = 0; link< NUM_LINKS; link++)
    write_reg( 0x700 +link, tem );

// storage of RX ICP cell, valid ICP cell with changes
temp = 0x0000;
write_reg( 0x0100, temp );
temp = 0x0000;
write_reg( 0x0101, temp );

// Rx Link Control
// don't enable IMA mode yet
for (group = 0; group<NUM_GROUPS; group++){
    for (i = 0; i < 2; i++) {

        link = i + group*2;
        if (link < 8){
            temp = 0x0000 | ( (read_reg( 0x00C0 +link) & 0xFF00));
            write_reg( 0x00C0 +link, temp);
        }
        else{
            temp = 0x0000 | ( (read_reg( 0x00C0+link-0x8) & 0x00FF));
            write_reg( 0x00C0 +link -0x8, temp);
        }

        // Set OAM label
        temp = 0x0001;
        write_reg( 0x00DD, temp );

        // Wait for cell delineation
        flag = 0;
        while (flag < 100)
        {
            // Cell Delineation Status
            temp = read_reg( 0x00E6);
            temp = (temp >> link) & 0x1;
            if (temp == 1) break;
            flag ++;
            Sleep (1); // 1 ms
        }
        if(flag == 100) printf("\nCell delin. error for link%x\n", link);
    }
}

for ( group = 0; group < NUM_GROUPS; group++)
```

```

{
    // Rx Reference link assignment
    link = group * 2;
    write_reg( 0x0209 + group, 0x0000|link);

    for (i = 0; i <2; i++)
    {
        //Assign link to RX recombiner group, don't enable recombination
        link = i + group * 2;
        if (link < 8){
            temp = (0x0000|group)|(( read_reg( 0x0201 +link) & 0xFFE0));
            write_reg( 0x201 +link, temp);
        }
        else{
            temp = ((0x0|group) <<8)|((read_reg(0x0201 +link- 0x8)&0xE0FF));
            write_reg( 0x201 +link -0x8, temp);
        }
    }

    Sleep(100);

    // enable RX IMA mode in Rx Link Control Reg
    if (link < 8){
        temp = 0x0040 | ( (read_reg( 0x00C0 +link) & 0xFF00));
        write_reg( 0x00C0 +link, temp);
    }
    else{
        temp = 0x4000 | ( (read_reg( 0x00C0 +link-0x8) & 0x00FF));
        write_reg( 0x00C0 +link-0x8, temp);
    }

    // Wait for IMA frame sync
    flag = 0;
    while (flag < 2000)
    {
        // IMA Sync Status
        temp = read_reg( 0x00E5);
        temp = (temp >> link) & 0x1;
        if (temp == 1) break;
        flag ++;
        Sleep (2);
    }
    if (flag == 2000) printf("\n IMA sync error on link %x.\n", link);

    // Enable recombination
    if (link < 8){
        temp = 0x0010|(( read_reg( 0x0201 +link) & 0xFFEF));
        write_reg( 0x0201 +link, temp);
    }
    else{

```

```

    temp = 0x1000 | ( (read_reg( 0x0201 +link-0x8) & 0xEFFF));
    write_reg( 0x0201 +link - 0x8, temp);
}
// Wait until recombination confirmed
flag=0;
while (flag < 100)
{
    // Rx Recombiner Status
    temp = read_reg( 0x02AD);
    temp = (temp >> link) & 0x1;
    if (temp == 1) break;
    flag ++;
    Sleep (150); // 150 ms
}
if(flag == 100) printf("\nRx recom. error on link.%x\n", link);

} //End link for_loop
} // End group for_loop

/*-----*
* Start Up
*-----*/

for ( group = 0; group < NUM_GROUPS; group++){

    // Set Rx PHY address      ### user changeable ###
    if (group < 4){
        temp = (0x0000|(group +1)) | ( (read_reg(0x0008 + group) & 0xFFE0));
        write_reg( 0x0008 + group, temp);
    }
    else{
        temp = (0x0|(group+1)<< 8) | ((read_reg(0x0008+group-0x4) & 0xE0FF));
        write_reg( 0x0008 + group - 0x4, temp);
    }
    // Enable RX Utopia (UTOPIA Output)
    temp = (0x0001<<group) | (read_reg( 0x0011) );
    write_reg( 0x0011, temp );

    // Start sending user cells on IMA mode links
    for ( i = 0; i<2; i++) {
        link = i + group * 2;
        if (link < 8){
            temp = 0x0040 | ( (read_reg( 0x0318 +link) & 0xFFBF));
            write_reg( 0x0318 +link, temp);
        }
        else{
            temp = 0x4000 | ( (read_reg( 0x0318 +link-8) & 0xBFFF));
            write_reg( 0x0318 +link -8, temp);
        }
    }
}

```

```
    }
    Sleep(100); //100 ms

    // Tx PHY address      ### user changeable ###
    if (group < 4){
        temp = (0x0|(group+1))|((read_reg(0x0048 + group) & 0xFFE0));
        write_reg( 0x0048 + group, temp);
    }
    else{
        temp = (0x0|(group+1)<< 8)|((read_reg(0x0048+group-0x4)&0xE0FF));
        write_reg( 0x0048 + group - 0x4, temp);
    }
    // Enable UTOPIA PHY TX direction
    temp = (0x0001<<group)|(read_reg( 0x0051)) ;
    write_reg( 0x0051, temp );
    } //end of group for_loop

    //printf("\n ***UP AND RUNNING***\n");
    Sleep(5);

    return;
} //end function Setup8g2l
```



**For more information about all Zarlink products
visit our Web Site at
www.zarlink.com**

Information relating to products and services furnished herein by Zarlink Semiconductor Inc. or its subsidiaries (collectively "Zarlink") is believed to be reliable. However, Zarlink assumes no liability for errors that may appear in this publication, or for liability otherwise arising from the application or use of any such information, product or service or for any infringement of patents or other intellectual property rights owned by third parties which may result from such application or use. Neither the supply of such information or purchase of product or service conveys any license, either express or implied, under patents or other intellectual property rights owned by Zarlink or licensed from third parties by Zarlink, whatsoever. Purchasers of products are also hereby notified that the use of product in certain ways or in combination with Zarlink, or non-Zarlink furnished goods or services may infringe patents or other intellectual property rights owned by Zarlink.

This publication is issued to provide information only and (unless agreed by Zarlink in writing) may not be used, applied or reproduced for any purpose nor form part of any order or contract nor to be regarded as a representation relating to the products or services concerned. The products, their specifications, services and other information appearing in this publication are subject to change by Zarlink without notice. No warranty or guarantee express or implied is made regarding the capability, performance or suitability of any product or service. Information concerning possible methods of use is provided as a guide only and does not constitute any guarantee that such methods of use will be satisfactory in a specific piece of equipment. It is the user's responsibility to fully determine the performance and suitability of any equipment using such information and to ensure that any publication or data used is up to date and has not been superseded. Manufacturing does not necessarily include testing of all functions or parameters. These products are not suitable for use in any medical products whose failure to perform may result in significant injury or death to the user. All products and materials are sold and services provided subject to Zarlink's conditions of sale which are available on request.

Purchase of Zarlink's I²C components conveys a licence under the Philips I²C Patent rights to use these components in and I²C System, provided that the system conforms to the I²C Standard Specification as defined by Philips.

Zarlink, ZL and the Zarlink Semiconductor logo are trademarks of Zarlink Semiconductor Inc.

Copyright Zarlink Semiconductor Inc. All Rights Reserved.

TECHNICAL DOCUMENTATION - NOT FOR RESALE
