Advanced **L**ogic **S**ynthesis for **E**lectronics
http://www.alse-fr.com

ALSE's **GEDEK** Register Interface Application Note

May 2009, ver 2009.05 **Description**

Introduction

This document present the protocol used by the Gedek for the register interface.

This protocol is used in both direction, ie it is used by the remote computer as well as the FPGA. All the register frames are sent/received using the data stream port, thus the commands are stored in a standard UDP Payload.

The distinction between a register access and a data stream is achieved by monitoring the UDP port used for the rx/tx frame.

The register port is configured in the FPGA with the RegisterPort constant.

In the reference design, we are using the port 0x04D3 for the register interface.

Register Frame Format

In order to access to the register interface, the user shall send the following UDP frame to the configured register port :

1st word of the UDP Payload

31..24	23.16	15..8	7..0
Addr	Command	0xFF	0x00

2nd word of the UDP Payload (optional – in case of write request)

31..0
Data to be written in the register @Addr

In the 1st word, Command can take the following values :

Command Value	**Command**
0x04	Read Register
0x08	Write Register

In the 1st word, Addr is the address of the register to be accessed.

Here is the equivalent C code for a read access to the register @0x01:

```
int sendbuf[2];
int regaddress = 0x01;
sendbuf[0] = (regaddress << 24) + 0x0004FF00;
n_sent = sendto( socket_id, sendbuf, 1*sizeof(int), 0, (struct sockaddr *)&server,
server_ln);
```

Here is the equivalent C code for a write access to the register @0xA7 with 0x12345678:

```
int sendbuf[2];
int regaddress = 0xA7;
sendbuf[0] = (regaddress << 24) + 0x0008FF00;
sendbug[1] = 0x12345678;
n_sent = sendto( socket_id, sendbuf, 2*sizeof(int), 0, (struct sockaddr *)&server,
server_ln);
```

Here is the decoding part of this frame in the VHDL reference design :

```
      Case ( UDPRxState ) Is
-- sIDLE
        When sIDLE =>
          If (UserRxUDP_DstPort = RegisterPort) Then

            -- Read Register
            If    ((UserRxUDP_Sop = '1') and (UserRxUDP_Data(23 Downto 16) = x"04")
and (UserRxUDP_Data(15 Downto 8) = x"00" or UserRxUDP_Data(15 Downto 8) = x"FF"))
Then --" Register Read "
              RegAddress          <= UserRxUDP_Data(31 Downto 24);
              report "Read Register";
              MasterAnswerWr      <= '1';
              MasterAnswerWrData  <= UserRxUDP_Data(31 Downto 24) & x"040000";
              UDPRxState          <= sReadRegister;

            -- Write Register
            ElsIf ((UserRxUDP_Sop = '1') and (UserRxUDP_Data(23 Downto 16) = x"08")
and (UserRxUDP_Data(15 Downto 8) = x"00" or UserRxUDP_Data(15 Downto 8) = x"FF"))
Then --" Register Write  "
              report "Write Register";
              RegAddress          <= UserRxUDP_Data(31 Downto 24);
              UDPRxState          <= sWriteRegister;

            -- Unknown !
            ElsIf ((UserRxUDP_Sop = '1') and (UserRxUDP_Data(15 Downto 8)  = x"00"))
Then
              UDPRxState          <= sUnRecognized;
            End If;
          End If;
```

Please note that :

- MasterAnswerWrData is handling the data which is sent to the remote host.
- MasterAnswerWr is acting as a MasterAnswerWrData Valid flag.
- UserRxUDP_Data is handling the Gedek received data
- UserRxUDP_Sop is the Start Of Packet bit coming from the Gedek
- UserRxUDP_Eop will be asserted during the last work of the current rx frame.
- UserRxUDP_DstPort is the destination port of the current rx frame.

The address of the register accessed is available in the RegAddress signal.

From this code, we see that the user will be able to treat the request in the sReadState or sWriteState.

The the sReadRegister can be coded like this :

```
-- sReadRegister
        When sReadRegister =>
          MasterAnswerWr       <= '1';
          MasterAnswerWrData   <= (Others => '0');
          UDPRxState           <= sIDLE;
          Case RegAddress Is
            When x"00" =>
              MasterAnswerWrData        <= Register00;
            When x"01" =>
              MasterAnswerWrData        <= Register01;
                                         ....
```

In the sWriteRegister, it is done in the same way except that we have to wait for the next valid received data from the Gedek :

```
-- sWriteRegister
        When sWriteRegister =>
          If (UserRxUDP_Dav = '1') Then
            UDPRxState     <= sIDLE;
            Case RegAddress Is

              When x"00" =>
                Register00        <= UserRxUDP_Data;

              When x"01" =>
                Register01        <= UserRxUDP_Data;
                                         ...
```

Sample C Code to Read then Write a Register

The following C code will send a read request to the Gedek in order to retrieve the register @0xA7, check the returned value then write the register @0xFF with a dummy value.

```
// THIS CODE IS CONFIDENTIAL AND CANNOT BE DISTRIBUTED
///////////////////////////////////////////////////
// Copyright   : ALSE - http://alse-fr.com
// Contact     : info@alse-fr.com
// Module      : gedek
// Description : Gedek main tool example
///////////////////////////////////////////////////
// Auteur      : G. JOLI
// Date        : May 2009
// Version     : 09.05
///////////////////////////////////////////////////
```

```
//
// 09.05 : Initial Version
//

#include <stdio.h>
#include     <sys/socket.h>
#ifdef __linux__
    #include <arpa/inet.h>
#else
    #include <cygwin/in.h>
    #include <arpa/inet.h>
#endif
#include <errno.h>

#define IP_ADDRESS  "192.168.1.18"
#define REG_PORT    0x04D3


int main()
{
  struct sockaddr_in server;
  int           socket_id     = 0;
  unsigned int  buffer[4]     = {0,0,0,0};
  int           n_sent        = 0;
  int           n_read        = 0;
  int           server_ln     = 0;
  int           rc            = 0;
  int           regaddress    = 0;


  printf("\n");
  printf("ALSE GEDEK Register Example Software\n");
  printf("Guillaume JOLI - gjoli@alse-fr.com - (c) ALSE'09\n\n");
  printf("Version 09.05 - May 04th 2009\n\n\n");

  printf("] Monitoring UDP Port 0x%04x for registers\n\n", REG_PORT);

  // Register Socket
  if( (socket_id = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0 )
  {
    perror("! Error while creating socket !\n");
    return(1);
  }

  memset((char*) &server, sizeof(server), 0 );
  server.sin_family       = AF_INET;                    // Used protocol: INET
  server.sin_addr.s_addr  = INADDR_ANY;                 // Any address can connect to
the socket
  server.sin_port         = htons(REG_PORT);            // Setup the probing port


  // Connect to the socket
  if ( (rc = bind(socket_id, (struct sockaddr *)&server, sizeof(server))) < 0 )
  {
    perror ("! Error while binding to the register socket !");
    return(rc);
  }
  // Initialize the server structure
  server.sin_addr.s_addr  = inet_addr(IP_ADDRESS);     // Send only data to the
target
  server_ln               = sizeof(server);


  // We are going to access to the register located @regaddress
  regaddress = 0xFF;
  srand(time(NULL));



  ////////////////////////////////////////////////////////////////////
  // Send the Read Request
  ////////////////////////////////////////////////////////////////////
  printf("] Sending the Read Request @%02x.  ", regaddress);
  buffer[0] = 0x0004FF00 + (regaddress << 24);

  n_sent = sendto( socket_id, buffer, 1*sizeof(int), 0, (struct sockaddr *)&server,
server_ln);

  if (n_sent > 0)
    printf("OK ! [ 0x%08x ]\n", buffer[0]);
  else
    perror("Error : ");


  ////////////////////////////////////////////////////////////////////
  // Receive the register value
  ////////////////////////////////////////////////////////////////////
  printf("] Waiting for the Register Value... \n");
  n_read = recvfrom( socket_id, buffer, 512*sizeof(int), 0, (struct sockaddr
*)&server, &server_ln);
  if ((buffer[0] & 0x00FFFF00) == 0x00040000)
  {
    printf("[%02x] <@%02x> = 0x%08x\n", (buffer[0] & 0xFF), (buffer[0] & 0xFF000000)
>> 24, buffer[1]);
  }
  else
  {
    printf("Error in Answer received : ");
    for (n_sent=0;n_sent<n_read/4;n_sent++)
    {
      printf("%08x ", buffer[n_sent]);
    }
    printf("\n");
  }



  ////////////////////////////////////////////////////////////////////
  // Send the Write Request
  ////////////////////////////////////////////////////////////////////
  printf("] Sending the Write Request @%02x. ", regaddress);
  regaddress = 0xFF;
  buffer[0] = 0x0008FF00 + (regaddress << 24);
  buffer[1] = rand();

  n_sent = sendto( socket_id, buffer, 2*sizeof(int), 0, (struct sockaddr *)&server,
server_ln);

  if (n_sent > 0)
    printf("OK ! [ 0x%08x | 0x%08x ]\n", buffer[0], buffer[1]);
  else
    perror("Error : ");


  ////////////////////////////////////////////////////////////////////
  // Send again the Read Request
  ////////////////////////////////////////////////////////////////////
  printf("] Sending the Read Request @%02x.  ", regaddress);
  buffer[0] = 0x0004FF00 + (regaddress << 24);

  n_sent = sendto( socket_id, buffer, 1*sizeof(int), 0, (struct sockaddr *)&server,
server_ln);

  if (n_sent > 0)
    printf("OK ! [ 0x%08x ]\n", buffer[0]);
  else
    perror("Error : ");


  ////////////////////////////////////////////////////////////////////
  // Receive the new register value
  ////////////////////////////////////////////////////////////////////
  printf("] Waiting for the Register Value...\n");
  n_read = recvfrom( socket_id, buffer, 512*sizeof(int), 0, (struct sockaddr
*)&server, &server_ln);
  if ((buffer[0] & 0x00FFFF00) == 0x00040000)
  {
    printf("[%02x] <@%02x> = 0x%08x\n", (buffer[0] & 0xFF), (buffer[0] & 0xFF000000)
>> 24, buffer[1]);
  }
  else
  {
    printf("Error in Answer received : ");
    for (n_sent=0;n_sent<n_read/4;n_sent++)
    {
      printf("%08x ", buffer[n_sent]);
    }
    printf("\n");
  }

  printf("] Done !\n");

  return 0;
}
```

Conclusion

With these informations, you should be able to use efficiently the register interface of the Gedek.
In case of any further question, contact ALSE at : ip_support@alse-fr.com