application note

V1190A/V1190B
V1290A/V1290N
VX1190A/VX1190B
VX1290A/VX1290N

FIRMWARE UPGRADE

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

The module houses a serial FLASH where the two Firmware versions of the FPGA are stored. The module is originally issued with the most recent firmware revision on both the “standard” and the “back up” version. Fig. 1 shows the FLASH content; please refer to the modules User Manual for further information.

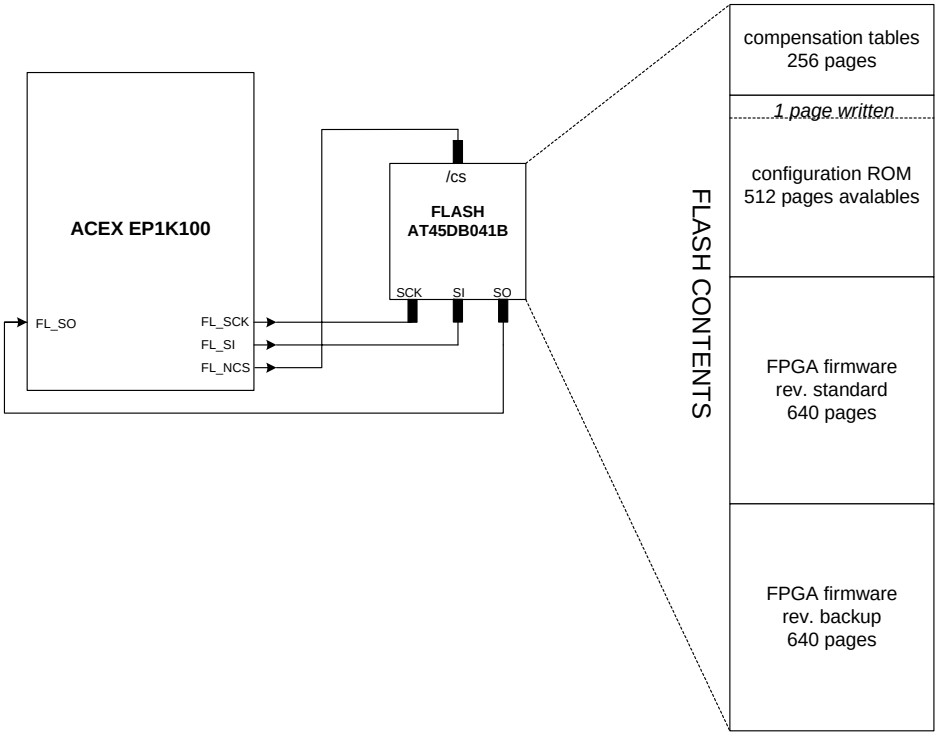

Fig. 1: FLASH Architecture

At Power On a microcontroller programs the FPGA with the Firmware revision selected via the J13 Jumper (default position: STD).

2 Firmware upgrade

The FPGA firmware can be updated via VME. The FLASH content is overwritten during this operation.
This operation is reserved for experienced Users: our advice is to overwrite only one firmware revison (for example the Standard) and to verify the correct module operation before overwriting the other one.
The following C example (cconf.c) shows how to upload a new firmware release into the Flash via VME.
The two most recent firmware releases are contained in the present package (**v1190corex.y.rbf** – *x.y* represents the firmware release).

```
/*****************************************************************************
  CCONF: CAEN Load FPGA Configuration File

  This program writes the configuration file (Altera RBF Format) into one
  of the two pages (STD and BCK) of the flash memory in the CAEN's modules.
  The flash memory is written via VME.
*****************************************************************************/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define FIRST_PAGE_STD    768    // first page of the copy STD
#define FIRST_PAGE_BCK    1408   // first page of the copy BCK


/*****************************************************************************/
/* MAIN
/*****************************************************************************/
void main(int argc,char *argv[])
{
  int finish,i;
  int pp, bp, bcnt, page;
  char filename [1000];
  char c;
  unsigned char pdw[264],pdr[264];
  FILE *cf;


  printf("\n\n");
  printf("*********************************************************\n");
  printf("* CAEN SpA - Front-End Division                         *\n");
  printf("* ----------------------------------------------------- *\n");
  printf("* ISP Write Flash for FPGA configuration                *\n");
  printf("* Version 1.0 (28/07/04)                                *\n");
  printf("*********************************************************\n\n");

  if (argc<2)
    {
    printf("\n\n");
```

```
  printf("Syntax: CCONF filename [copy] \n");
  printf("   where copy is 0 for Standard and 1 for Backup\n\n");
  printf("Press SPACE to continue or any other key to exit");
  c = getch();
  if (c != ' ')
    exit(-1);

  printf("\n\nEnter RBF file name : ");
  scanf("%s",filename);
  }
else
  strcpy(filename,argv[1]);

page=0;
if (argc == 3)
  sscanf(argv[2],"%d",&page);

// open the configuration file
cf=fopen(filename,"rb");
if(cf==NULL)
  {
  printf("\n\nCan't open file %s\n",filename);
  exit(-1);
  }

  printf("\n\nLoading file %s\n",filename);
  if (page==0)
    printf("Writing copy STD of the firmware\n");
  else
    printf("Writing copy BCK of the firmware\n");

  bcnt=0;            // byte counter
  if (page==0)
    pp=FIRST_PAGE_STD; // page pointer
  else
    pp=FIRST_PAGE_BCK; // page pointer

  bp=0;              // byte pointer in the page
  finish=0;          // it goes high when eof is found

  // while loop
  while(!finish)
  {
    c=(unsigned char)fgetc(cf);  // read one byte from file
    // mirror byte (lsb becomes msb)
    pdw[bp]=0;
    for(i=0; i<8; i++)
      if(c & (1<<i))
        pdw[bp] = pdw[bp] | (0x80>>i);
    bp++;
    bcnt++;
    if(feof(cf))
      finish=1;

    //write and verify a page
```

```
      if((bp==264) || finish)
        {
          printf(".");

          write_flash_page(pdw,pp); // write page
          read_flash_page(pdr,pp);  // read page
          for(i=0; i<bp; i++)       // verify page
            if(pdr[i] != pdw[i])
              {
              printf("\n\nFlash writing failure (byte %d of page %d)!",i,pp);
              printf("\nFirmware not loaded!",i,pp);
              exit(-1);
              }
          bp=0;
          pp++;
        }
      }  // end of while loop

    fclose(cf);

    printf("\nFirmware loaded without errors. Written %d bytes\n",bcnt);
}
```

3 Read/write procedures

Two VME read/write procedures, referred to a page of the FLASH, are shown in the following (**write_flash_page** and **read_flash_page**); actually such procedures depend on the development enviroment and on the used VME controller.

```
/*****************************************************************************/
/*              FLASH memory accesses adresses & opcodes                     */
/*****************************************************************************/
#define MAIN_MEM_PAGE_READ          0x00D2
#define MAIN_MEM_PAGE_PROG_TH_BUF1  0x0082
#define PAGE_ERASE                  0x0081

#define SEL_FLASH                   0x1032
#define FLASH_MEM                   0x1034

unsigned long v1190_base_address;   /* base address of the board */
```

```
/*****************************************************************************/
/*                            WRITE_FLASH_PAGE                               */
/*---------------------------------------------------------------------------*/
/* parameters     :     pagenum: number of page to write                       */
/*                      page   : page to write                                */
/*---------------------------------------------------------------------------*/
/* write a flash page                                                        */
/*****************************************************************************/
void write_flash_page(unsigned char* page, unsigned long pagenum)
{
  unsigned long  flash_addr,data;
  unsigned char  addr0,addr1,addr2;
  unsigned short i;

  flash_addr = pagenum<<9;
  addr0 = (unsigned char) flash_addr;
  addr1 = (unsigned char)(flash_addr>>8);
  addr2 = (unsigned char)(flash_addr>>16);

  /* enable flash (NCS = 0) */
  data = 0;
  vme_write(v1190_base_address + SEL_FLASH, data, A32, D16);

  /* write opcode */
  data = MAIN_MEM_PAGE_PROG_TH_BUF1;
  vme_write(v1190_base_address + FLASH_MEM, data, A32, D16);

  /* write address */
  data = addr2;
  vme_write(v1190_base_address + FLASH_MEM, data, A32, D16);
  data = addr1;
  vme_write(v1190_base_address + FLASH_MEM, data, A32, D16);
  data = addr0;
  vme_write(v1190_base_address + FLASH_MEM, data, A32, D16);

  /* write flash page (264 bytes for page, 256 used) */
  for (i=0; i<264; i++)
  {
    data = page[i];
    vme_write(v1190_base_address + FLASH_MEM, data, A32, D16);
  }

  /* disable flash (NCS = 1) */
  data = 1;
  vme_write(v1190_base_address + SEL_FLASH, data, A32, D16);

  /* wait 20ms (max time required by the flash to complete the writing) */
  delay(20);
}
```

```
/*****************************************************************************/
/*                            READ_FLASH_PAGE                                */
/*---------------------------------------------------------------------------*/
/* parameters     :     pagenum: number of page to read                      */
/*                      page   : read page                                   */
/*---------------------------------------------------------------------------*/
/* read a flash page                                                         */
/*****************************************************************************/
void read_flash_page(unsigned char* page, unsigned long pagenum)
{
  unsigned long  flash_addr,data;
  unsigned char  addr0,addr1,addr2;
  unsigned short i;

  flash_addr = pagenum<<9;
  addr0 = (unsigned char)flash_addr;
  addr1 = (unsigned char)(flash_addr>>8);
  addr2 = (unsigned char)(flash_addr>>16);

  /* enable flash (NCS = 0) */
  data = 0;
  vme_write(v1190_base_address + SEL_FLASH, data, A32, D16);

  /* write opcode */
  data = MAIN_MEM_PAGE_READ;
  vme_write(v1190_base_address + FLASH_MEM, data, A32, D16);

  /* write address */
  data = addr2;
  vme_write(v1190_base_address + FLASH_MEM, data, A32, D16);
  data = addr1;
  vme_write(v1190_base_address + FLASH_MEM, data, A32, D16);
  data = addr0;
  vme_write(v1190_base_address + FLASH_MEM, data, A32, D16);

  /* additional don't care bytes */
  data = 0;
  for (i=0; i<4; i++)
  {
    vme_write(v1190_base_address + FLASH_MEM, data, A32, D16);
  }

  /* read flash page (264 bytes for page, 256 used) */
  for (i=0; i<264; i++)
  {
    vme_read(v1190_base_address + FLASH_MEM, &data, A32, D32)
    page[i] = data;
  }

  /* disable flash (NCS = 1) */
  data = 1;
  vme_write(v1190_base_address + SEL_FLASH, data, A32, D16);
}
```