
TN8000.02

Technical Note

XE8801A/05A Programming

Contents

1. Introduction	3
2. Abbreviations	3
3. Physical programming interface	3
4. Programming parameters	4
4.1. Operating conditions	4
4.2. Timing Parameters.....	4
4.3. Power-up and down sequences	7
5. Programming the whole MTP array using default settings	8
5.1. Programming sequences.....	9
5.1.1. Introduction	9
5.1.2. Define constante.....	9
5.1.3. Define macro.....	9
5.2. Programming the whole array.....	13
5.3. Error description.....	14
5.4. Checksum computing	15
Table 1: Abbreviation description	3
Table 2: Pins in flash programming mode and their functions	3
Table 3: programming conditions	4
Table 4 Clock timing constraints.....	4
Table 5 high voltage pulses timing constraints.....	5
Table 6 Sequence description for the shift_signature and cycle_cr_ck	9
Table 7: programming errors	14
Figure 1: Top view in testmode	4
Figure 2 : Timing diagram for crck clock.....	5
Figure 3 : Timing diagram for ptck clock.....	5
Figure 4 : Timing diagram for ptck clock during the erase verify.....	5
Figure 5 : Timing diagram for shift_instruction	5
Figure 6 : Timing diagram write_cr_normal	6
Figure 7 : Timing diagram write_cr	6
Figure 8 : Timing diagram for pulse_high_voltage, for (A) short and (B) long pulses.....	6
Figure 9 : Timing diagram for the read_fault	6
Figure 10 : Timing diagram for the checksum	7
Figure 11: Timing diagram for the lock_test	7
Figure 12: Timing diagram for the power-on	7
Figure 13: Timing diagram for the power-down.....	7
Figure 14 : Complete programming flow chart	8
Figure 15 : Example of sequence for the shift_signature following by cycle_crck	9

1. Introduction

This document describes the programming flow of the XE8801/01A, and 05/05A products.

Firstly, the conditions and parameters are described. Then the programming flow will be given followed by the main routines implemented, and finally the correct usage of both to obtain a correctly programmed and verified device.

2. Abbreviations

Table 1 shows the different abbreviations used in this document.

Abbreviation name	Description
MTP	Multiple Time Programmable
Crck	Coolrisc clock
Ptck	Peripheral test clock
HV	High Voltage
Testck	serial clock for shift instruction
Testin	serial input for shift instruction
Testout	serial output for shift instruction

Table 1: Abbreviation description

3. Physical programming interface.

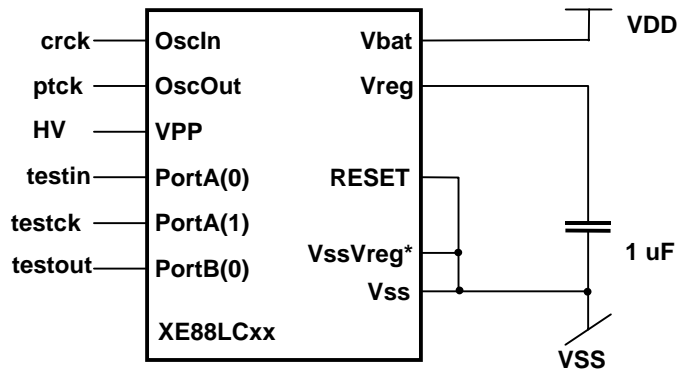
The CoolRisc must be in test mode to perform the programming. Table 2 shows the function of the pins used in this test mode.

PAD name	Type	Function prog mode	in value	Signal type
Vbat	Power supply input	Vdd	Vdd	Static
Vss	Power supply input	Vss	Vss	Static
RESET	Digital input under Vbat	Reset	Vss	Static
Vreg	Power supply output	Vreg (capacitance = 1 uF)		
OscIn	Digital input under Vbat	Crck	Vss or Vdd	dynamic
OscOut	Digital input under Vbat	Ptck	Vss or Vdd	dynamic
VPP	Power supply input	HV	Vdd or Vddt or Vddhigh	dynamic
PortA(0)	Digital input under Vbat	Testin	Vss or Vdd	dynamic
PortA(1)	Digital input under Vbat	Testck	Vss or Vdd	dynamic
PortB(0)	Digital output under Vbat	Testout	Vss or Vdd	dynamic

Table 2: Pins in flash programming mode and their functions

The PortB(0) is used as serial output to check programming results.

Other input pins of the device should ideally be tied to either the Vdd or Vss potential. This prevents currents in input buffers.



* : VssVreg pin is only available on some packages.

Figure 1: top view in testmode

4. Programming parameters.

4.1. OPERATING CONDITIONS

The **VPP** pad in program mode takes the value of Vdd or Vddt or Vddhigh following the programming state. (see timing diagram in the chapter 4.2 Timing Parameters)

Table 3 specifies the programming conditions of the device.

parameter	Min	Max	Unit	Remark
Vdd	4.5	5.5	V	
Vddt	Vdd+2.0	Vdd+2.5	V	
VddHigh	11.55	11.65	V	Iprog ≤ 30 mA
VREG capacitance	0.9	1.1	μF	
Temperature	10	40	Deg. Celcius	Typical = 25

Table 3: programming conditions

4.2. TIMING PARAMETERS

This chapter describes the timing constraints for device programming. The Table 4 shows all clock timing constraints and the Table 5 shows the high voltage pulses timing constraints.

Parameter	Description	min	max	unit
ta	Testin stable before rise of testck	50		ns
tb	Testin stable after rise of testck	50		ns
tr	Testck rise time		10	ns
tf	Testck fall time		10	ns
th	Testck high	125		ns
tl	Testck low	125		ns
tc	Crck and ptck clock high/low	800		ns
trfc	Crck and ptck clock rise/fall		10	ns
tff	Fast ptck clock during erase verify	115	125	ns

Table 4 Clock timing constraints

Parameter	Description	min	max	unit
trh	Rise time	0.5	3	us
tfh	Fall time	0.5	3	us
tsh	First short pulse duration	9	11	us
	Next short pulse duration	64	77	us
tlh	Long pulse duration	0.45	0.55	S

Table 5 high voltage pulses timing constraints

Figure 2 to Figure 11 show the timing diagrams related to the low-level macros described in 5.1.3.

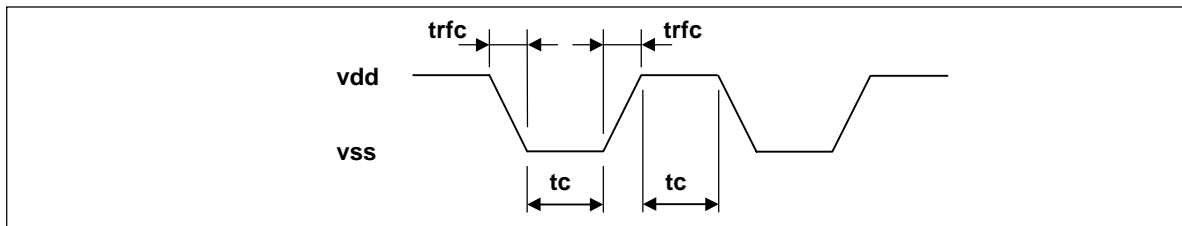


Figure 2 : Timing diagram for crck clock

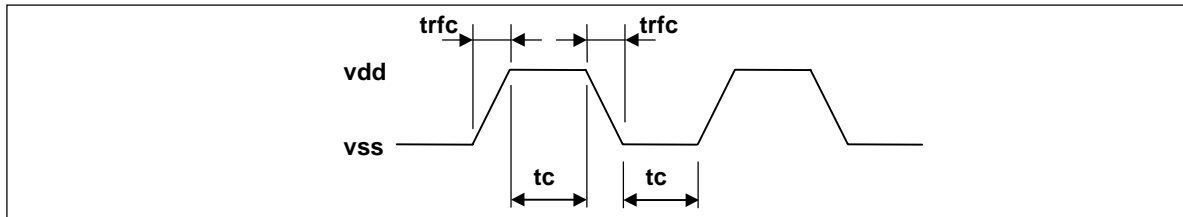


Figure 3 : Timing diagram for ptck clock

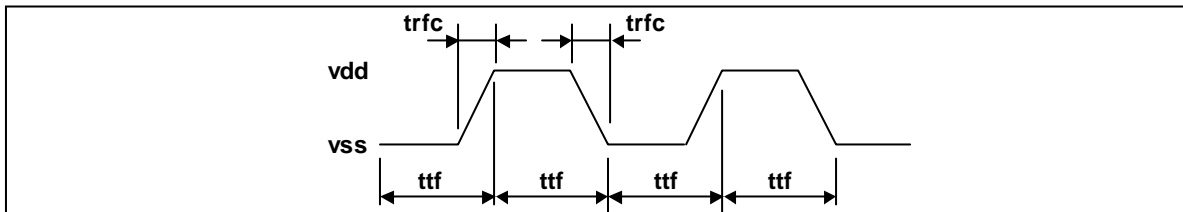


Figure 4 : Timing diagram for ptck clock during the erase verify

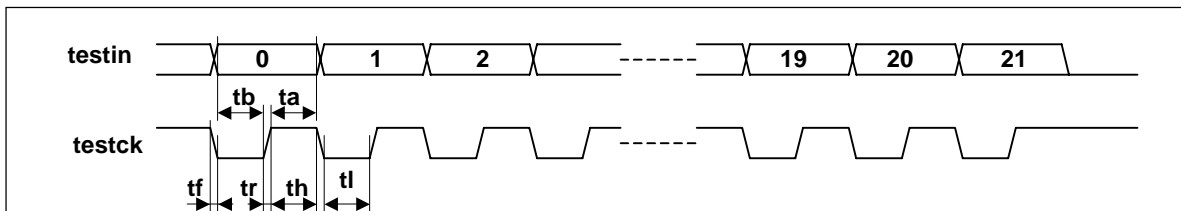


Figure 5 : Timing diagram for shift_instruction

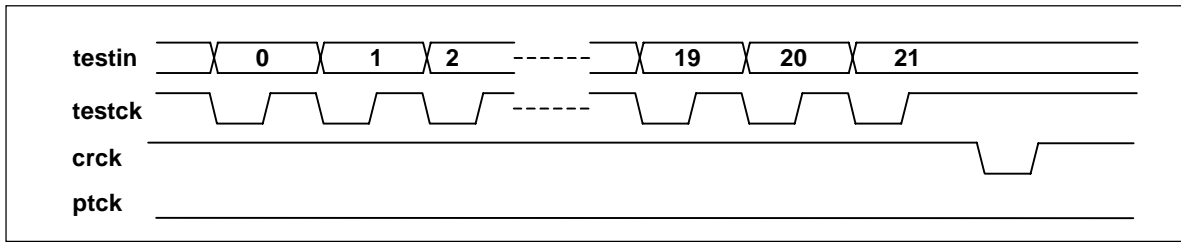


Figure 6 : Timing diagram write_cr_normal

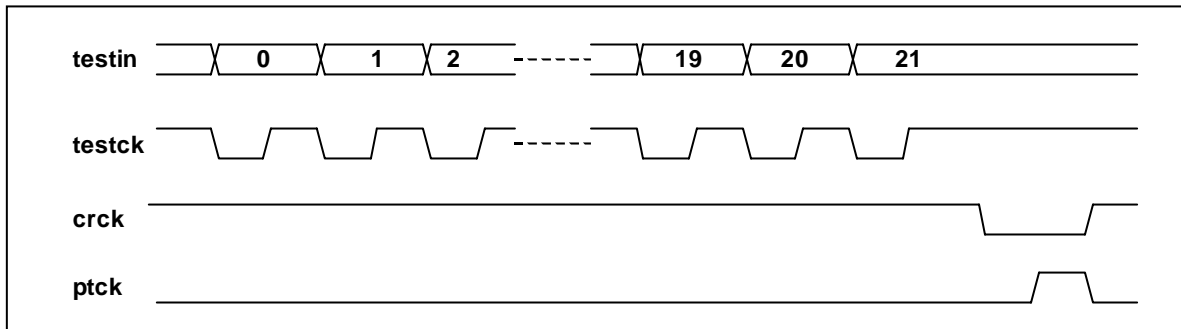


Figure 7 : Timing diagram write_cr

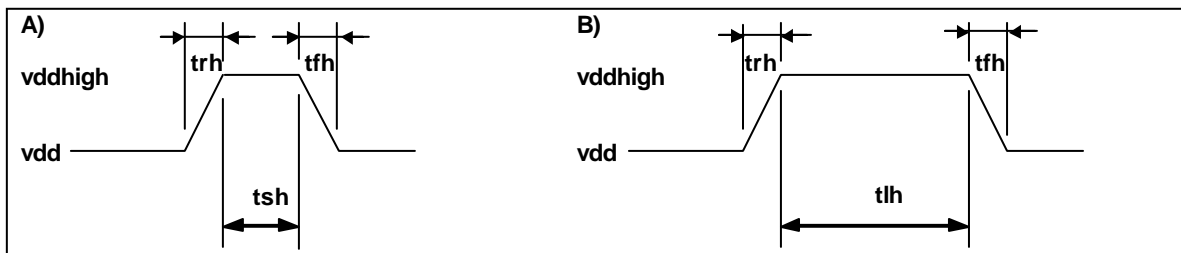


Figure 8 : Timing diagram for pulse_high_voltage, for (A) short and (B) long pulses

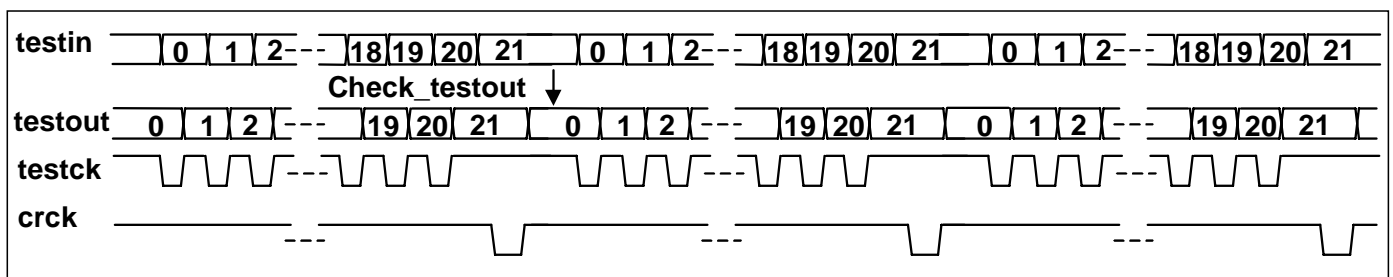


Figure 9 : Timing diagram for the read_fault

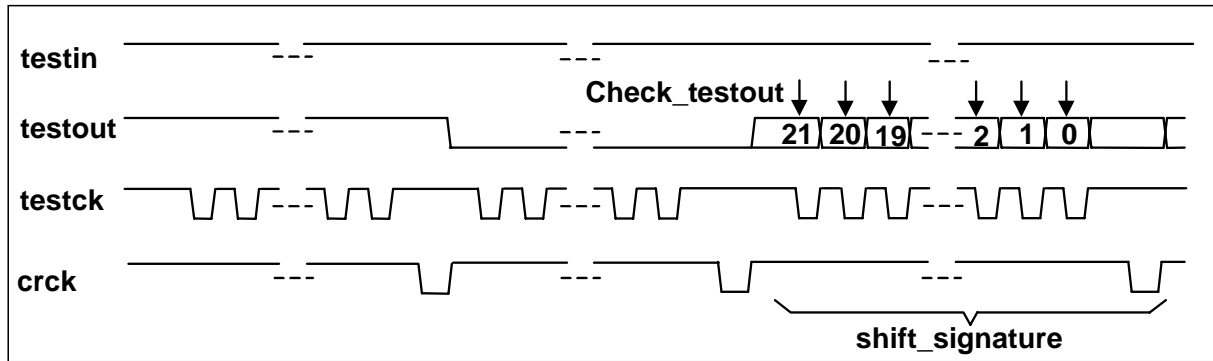


Figure 10 : Timing diagram for the checksum

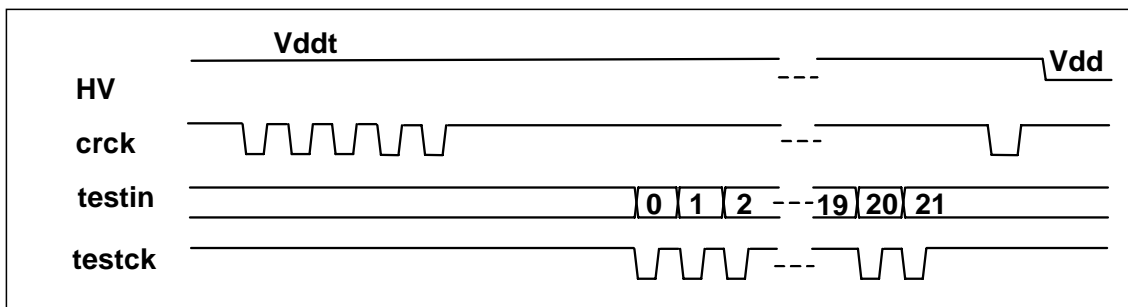


Figure 11: Timing diagram for the lock_test

4.3. POWER-UP AND DOWN SEQUENCES

Figure 12 and Figure 13 show the the timing diagrams related to the low-level macros power-on and power-down described in 5.1.3

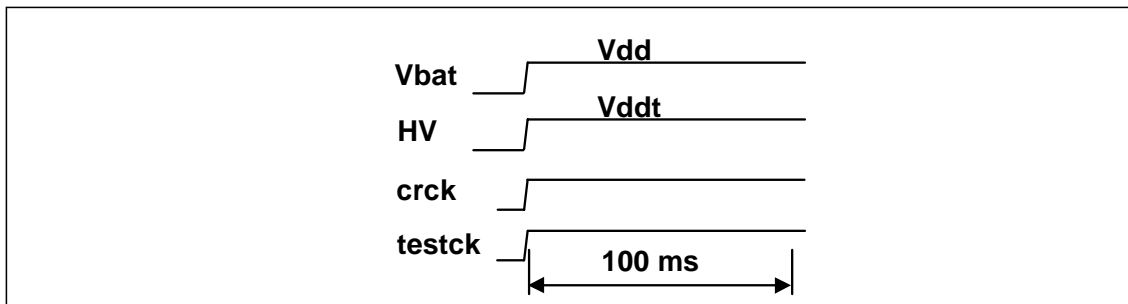


Figure 12: Timing diagram for the power-on

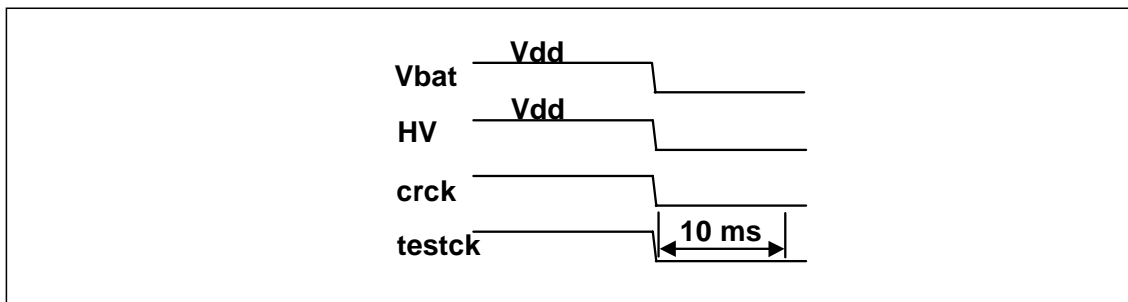


Figure 13: Timing diagram for the power-down

5. PROGRAMMING THE WHOLE MTP ARRAY USING DEFAULT SETTINGS

The programming process of the whole flash array using default settings runs as follows (check algorithm at the end of this chapter for detailed operation and up-to-date algorithm):

1. The programming procedure starts with a chip power-on sequence. (see chapter 4.3)
2. The programmer locks the chip in test mode (see macro lock_test, chapter 5.1.3).
3. The programmer erases and prepares the chip to receive data (see macro erase, chapter 5.1.3).
4. The programmer writes the data in the device memory (see macro write_data, chapter 5.1.3).
5. The programmer stops and restarts with a chip power-down and power-on. (see chapter 4.3)
6. The programmer verifies the complete programming result with the checksum (see the checksum macro, chapter 5.1.3).
7. The programming procedure finishes by power-down. (see chapter 4.3)

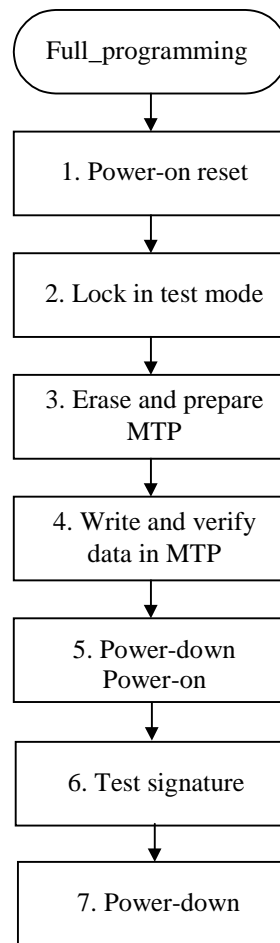


Figure 14 : Complete programming flow chart

5.1. PROGRAMMING SEQUENCES

5.1.1. Introduction

In the following meta language sequences, we use procedural macro definitions to describe the programming algorithm. Basic 'set' pattern lines represent the values put on the crck, testck, testin, ptck and high_voltage lines.

Function 'check_testout(x)' is a comparison of testout pin with the parameter 'x'. The variable ctrl2reg is a list of values for programming registers during the writing of the data.

5.1.2. Define constant

```
RegEEP = 0b00111000
RegEEP1 = 0b00111001
RegEEP2 = 0b00111010
RegEEP3 = 0b00111011

ctrl2reg{0} = 11101111
ctrl2reg{1} = 11101101
ctrl2reg{2} = 11101110
ctrl2reg{3} = 11101100
ctrl2reg{4} = 11101000
ctrl2regblk = 10101000
```

5.1.3. Define macro

Most timings are not expressed in the macros, only the sequences are described. Please refer to the timing diagrams of chapter 4.2 for reference.

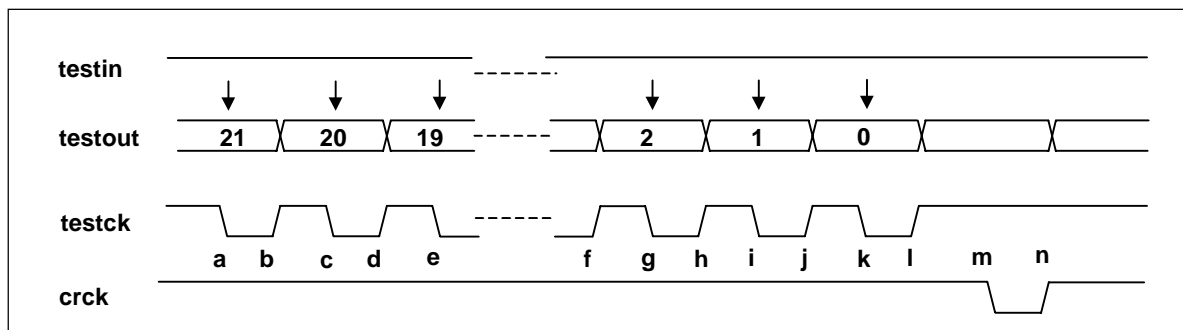


Figure 15 : Example of sequence for the shift_signature following by cycle_crck

Point	function	Description
a	set(1 0 1 0 vdd) and check_testout	set to vss testck and read testout (signature [21])
b	set(1 1 1 0 vdd)	set to vdd testck
C	set(1 0 1 0 vdd) and check_testout	set to vss testck and read testout (signature [20])
D	set(1 1 1 0 vdd)	set to vdd testck
E	set(1 0 1 0 vdd) and check_testout	set to vss testck and read testout (signature [19])
F	set(1 1 1 0 vdd)	set to vdd testck
G	set(1 0 1 0 vdd) and check_testout	set to vss testck and read testout (signature [2])
H	set(1 1 1 0 vdd)	set to vdd testck
I	set(1 0 1 0 vdd) and check_testout	set to vss testck and read testout (signature [1])
J	set(1 1 1 0 vdd)	set to vdd testck
K	set(1 0 1 0 vdd) and check_testout	set to vss testck and read testout (signature [0])
L	set(1 1 1 0 vdd)	set to vdd testck
M	set(0 1 1 0 vdd)	set to vss crck
N	set(1 1 1 0 vdd)	set to vdd crck

Table 6 Sequence description for the shift_signature and cycle_cr_ck

```
macro set(crck; testck; testin; ptck; vpp)
```

sets the corresponding pin to Vss if value is 0 or Vdd if value is 1. For the crck, testck, testin, and ptck signals.
 Sets the Vpp pin to the specified value (either Vdd, Vddt or VddHigh).

macro wait(value)

Waits for the specified time with all input signals frozen in their current state.

macro check_testout(status)

returns 0 or 1 depending on the testout line potential

0 means an error, the value read on testout is not the same that the value of status(status is 0 and potential on testout is vdd or status is 1 and potential on testout is vss)

1 means no error(status is 0 and potential is vss or status is 1 and potential is vdd)

macro cycle_crck(nbcycles) ; Toggle CoolRISC clock nbcycles times

repeat nbcycles

set(0 1 0 0 vdd)

set(1 1 0 0 vdd)

end repeat

end macro

macro cycle_ptck(nbcycles) ;Toggle periperal test clock nbcycles times

repeat nbcycles

set(1 1 0 1 vdd)

set(1 1 0 0 vdd)

end repeat

end macro

macro cycle_crck_ptck(nbcycles) ;Toggle CoolRISC clock and periperal test clock nbcycles times

repeat nbcycles

set(0 1 0 0 vdd)

set(0 1 0 1 vdd)

set(1 1 0 0 vdd)

end repeat

end macro

macro shift_instruction(inst(21:0)) ; Repeat shift bit to input full

for index=0 to 21 ; 22 bits intruction

set(1 0 inst(index) 0 vdd)

set(1 1 inst(index) 0 vdd)

end for

end macro shift_instruction

macro write_cr (address(7:0), data(7:0))

shift_instruction(000000 not(data) not(address))

cycle_crck_ptck(1)

end macro

macro write_cr_normal (address(7:0), data(7:0))

shift_instruction(000000 not(data) not(address))

cycle_crck(1)

end macro

function read_fault (address(7:0)) ; returns 0 or 1

define test_result: boolean (0 or 1)

shift_instruction(00010010101110 not(address))

cycle_crck(1)

test_result = check_testout(0)

shift_instruction(00010010101110 not(address))

cycle_crck(1)

shift_instruction(00010010101110 not(address))

cycle_crck(1)

return(test_result)

end macro

macro lock_test ; Got to test mode, then lock it

define inst: instruction (22 bits wide) ; set Vpp to Vddt

```

repeat 5
    set(0 1 0 0 vddt)
    set(1 1 0 0 vddt)
end repeat
inst=(000000 not(0x80) not(0x19))
for index=0 to 21 ; 22 bits instruction
    set(1 0 inst(index) 0 vddt)
    set(1 1 inst(index) 0 vddt)
end for
set(0 1 0 0 vddt)
set(1 1 0 0 vddt)
set(1 1 0 0 vdd) ; set Vpp to Vdd
end macro

macro shift_signature
for index=21 to 0
    set(1 0 1 0 vdd)
    signature(index)=check_testout(1) ; signature(index) = testout
    set(1 1 1 0 vdd)
end for

if signature is not equal at the expected value given by programmer, then device is
defective, see chapter 5.3 Error4
end macro

macro signature
shift_instruction(11111111111111111111)
cycle_crck(1)
shift_signature
cycle_crck(1)
end macro

macro checksum
shift_instruction(111010000000000000000001)
cycle_crck(1)
shift_instruction(0010111110111111111111)
cycle_crck(1)
shift_instruction(11111111111111111111)
cycle_crck(1)
for address = 0 to 8191
    if check_testout(0) then jump to signature
    shift_instruction(11111111111111111111)
    cycle_crck(1)
end for
signature
end macro

macro pulse_high_voltage (duration) ; Put high voltage on VPP
set(1 1 0 0 vddhigh)
wait(duration)
set(1 1 0 0 vdd)
end macro

macro wait_cr_short
define inst: short instruction (9 bits wide)
inst = (000100101)
for index=0 to 8 ; 9 bits
    set(1 0 inst(index) 0 vdd)
    set(1 1 inst(index) 0 vdd)
end for
cycle_crck(1)
end macro

```

```

macro erase
  define ok : boolean (0=false, 1=true)

  erase_again:
    write_cr_normal (0x1D, 00110000)
    wait(100 ms)
    write_cr(RegEEP, 00001000)
    write_cr(RegEEP2, 00000000)
    write_cr(RegEEP2, 00000000)
    write_cr(RegEEP3, 00000000)
    write_cr(RegEEP3, 00000000)
    write_cr(RegEEP3, 00000000)
    write_cr_normal(RegEEP1, ctrl2reg{4})
    write_cr(RegEEP1, ctrl2reg{4})
    wait_cr_short
    cycle_ptck(1)
    pulse_high_voltage(500 ms) ;tlh parameter
    cycle_ptck(2)
    pulse_high_voltage(500 ms) ;tlh parameter

  write_blocking: ; write blocking bits
    write_cr_normal(RegEEP1,ctrl2regblk)
    write_cr(RegEEP1,ctrl2regblk)
    write_cr(RegEEP,00001110)
    wait_cr_short
    cycle_ptck(1)
    for address = 0 to 8191
      write_cr(RegEEP2, address_LSB)
      write_cr(RegEEP2, address_MSB)
      wait_cr_short
      for counter = 0 to 3
        cycle_ptck(1)
        pulse_high_voltage(70 us) ; parameter tsh (always long tsh)
        cycle_ptck(4)
      end for
    end for

    ok = read_fault(RegEEP) ; check error bit
    if (not ok) then jump to write_blocking ; *** If not successful in 12 times,
    device is defective, see chapter 5.3 Error1
  ***

  write_cr_normal(0x1D, 00100000)
  wait(500 ms)
  write_cr(RegEEP,00000010)
  for address = 0 to 8191 ; check blocking bits
    write_cr(RegEEP2, address_LSB)
    write_cr(RegEEP2, address_MSB)
    wait_cr_short
    cycle_ptck(2) ; must be realised with ttf timing
    ; see Table 4 and Figure 4
  end for

  ok = read_fault(RegEEP) ; check error bit
  if (not ok) then jump to erase_again ; *** If not successful in 3 times,
  device is defective, see chapter 5.3 Error2 ***
end macro

```

```

macro write_data
  define ok : boolean (0=false, 1=true)

  data_load:
    write_cr_normal(0x1D, 00110000)
    wait(100 ms) ; 100 ms

  data_again:
    write_cr(RegEEP, 01100000)
    write_cr_normal(RegEEP1,ctrl2reg{0})
    write_cr(RegEEP1,ctrl2reg{0})
    for address = 0 to 8191 ; write instructions in memory
      write_cr(RegEEP2, address[7..0])
      write_cr(RegEEP2, address[15..8])
      write_cr(RegEEP3, instruction_data(address)[7..0])
      write_cr(RegEEP3, instruction_data(address)[15..8])
      write_cr(RegEEP3, 00 instruction_data(address)[21..16])
      for counter = 0 to 7
        if counter < 5
          write_cr(RegEEP1,ctrl2reg{counter})
          wait_cr_short
        end if
        cycle_ptck(1)
        if counter = 0
          pulse_high_voltage(10 us) ; parameter tsh (short tsh)
        else
          pulse_high_voltage(70 us) ; parameter tsh (long tsh)
        end if
        cycle_ptck(4)
      end for
    end for

    ok = read_fault(RegEEP) ; check error bit
    if (not ok) then jump to data_again; *** If not successful in 3 times, ignore
    error bit and quit macro ***
  end macro

macro test_signature
  lock_test
  checksum
end macro

```

5.2. PROGRAMMING THE WHOLE ARRAY

This part describes the full sequence needed to programme the device by using the macro instruction.

```

sequence full_programming

  power-on ; power-on the IC

  lock_test ; initialize programming mode

  erase ; erase and prepare device

  write_data ; write and verify data

  power-down

  power-on ; power-on the IC

  test_signature

  power-down

end sequence

```

5.3. ERROR DESCRIPTION

Table 7 shows the different errors that arrive in the full programming sequence.

Error No	Source macro	Description	Comment
Error1	erase	Data erase is incorrect (error type 1)	
Error2	erase	Data erase is incorrect (error type 2)	
Error4	checksum	Checksum is incorrect after programming	

Table 7: programming errors

5.4. CHECKSUM COMPUTING

This chapter shows the C program that calculates the signature to be compared with the result of the shift_signature macro.

```

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

FILE *f_ptr;
long int bist; // Contains the CheckSum

void XmxCheckSum (long data){
    int j;
    long int sum, bist17, bist6, data_lsb, bist_lsb;
    for (j = 0; j < 22; j++){
        // invert the incoming bit
        // data starts from lsb to msb
        // data % 2 = data modulo 2 = Division reminder (data/2)
        if (data % 2 == 0)
            data_lsb = 1;
        else
            data_lsb = 0;
        bist17 = (bist >> 17) % 2;
        bist6 = (bist >> 6) % 2;
        bist = bist * 2; // shift left
        bist = bist % (1 << 18); // just keep remainder
        sum = bist17 + bist6 + data_lsb; // XOR function
        if ((sum == 1) + (sum == 3)) {
            bist_lsb = 1;
        }
        else {
            bist_lsb = 0;
        }
        bist = bist + bist_lsb;
        data = data / 2;
    }
}

int main(int argc, char *argv[]){
    int i;
    long int address, data;

    if(argc == 2){
        f_ptr = fopen(argv[1], "r");
        if(f_ptr == NULL) {
            perror("File Read or Write error");
            exit(1);
        }
        data = 0;
        bist = 0;
        XmxCheckSum(data);
        while (fscanf(f_ptr, "%x %x", &address, &data) == 2){
            XmxCheckSum(data);
        }
        printf(" The program signature is (in hexa) : 0x%x \n", bist);
        fclose(f_ptr);
    }
    else{
        printf(" SYNTAX ERROR : XE8000CheckSum filename \n");
        exit(1);
    }
    return 0;
}

```

© Semtech 2006

All rights reserved. Reproduction in whole or in part is prohibited without the prior written consent of the copyright owner. The information presented in this document does not form part of any quotation or contract, is believed to be accurate and reliable and may be changed without notice. No liability will be accepted by the publisher for any consequence of its use. Publication thereof does not convey nor imply any license under patent or other industrial or intellectual property rights. Semtech. assumes no responsibility or liability whatsoever for any failure or unexpected operation resulting from misuse, neglect improper installation, repair or improper handling or unusual physical or electrical stress including, but not limited to, exposure to parameters beyond the specified maximum ratings or operation outside the specified range.

SEMTECH PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF SEMTECH PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE UNDERTAKEN SOLELY AT THE CUSTOMER'S OWN RISK. Should a customer purchase or use Semtech products for any such unauthorized application, the customer shall indemnify and hold Semtech and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs damages and attorney fees which could arise.

Contact Information

Taiwan	Tel: 886-2-2748-3380 Fax: 886-2-2748-3390	Switzerland	Tel: 41-32-729-4000 Fax: 41-32-729-4001
Korea	Tel: 82-2-527-4377 Fax: 82-2-527-4376	United Kingdom	Tel: 44-1794-527-600 Fax: 44-1794-527-601
Shanghai	Tel: 86-21-6391-0830 Fax: 86-21-6391-0831	France	Tel: 33-(0)169-28-22-00 Fax: 33-(0)169-28-12-98
Japan	Tel: 81-3-6408-0950 Fax: 81-3-6408-0951	Germany	Tel: 49-(0)8161-140-123 Fax: 49-(0)8161-140-124

Semtech International AG is a wholly-owned subsidiary of Semtech Corporation, which has its headquarters in the U.S.A