APPLICATION NOTE

ABSTRACT
This application note describes the three methods that can be used to program the FLASH code memory of the XA-G49 family of microcontrollers. It discusses in detail the operation of the In-System Programming (ISP) capability which allows these microcontrollers to be programmed while mounted in the end product. These microcontrollers also have an In-Application Programming (IAP) capability which allows them to be programmed under firmware control of the embedded application. This capability is also described.

AN716
In-circuit programming of the XA-G49 microcontroller

1999 Sep 30

Philips
Semiconductors

PHILIPS

In-circuit programming of the XA-G49 microcontroller AN716

INTRODUCTION

This document gives a brief list of features for the XA-G49 family of microcontrollers with FLASH memory, and describes several ways that the FLASH memory can be programmed.

XA-G49 FEATURES

- XA CPU
- 64 kB FLASH EPROM
- FLASH EPROM is sectored to allow the user to erase and reprogram sectors
- 2048 bytes internal RAM
- 2 kB masked BOOTROM for In-System Programming of the FLASH EPROM
- User callable BOOTROM subroutines for FLASH erase and programming
- Can automatically run user program or BOOTROM program at power-up
- Four 8-bit I/O ports
- Two full-duplex UART
- Three standard 16-bit timer/counters (Timer 0, Timer 1 and Timer 2)
- Idle and power-down modes
- Two external interrupt inputs
- Three security bits
- Fully static operation: 0 to 30 Mhz
- Packages: 44-pin PLCC, 44-pin QFP

The XA-G49 series of microcontrollers contain user programmable FLASH EPROM. The FLASH EPROM is organized into sectors of 8 kB or 16 kB. Individual sectors can be erased and reprogrammed.

The BOOT ROM overlays the program flash memory space at the top 2 kB address space, from F800H to FFFFH. The BOOT ROM may be turned off by SFR AUXR1 bit 7, and the upper 2 kB of FLASH memory is accessible by the user. The sectors are organized as follows:

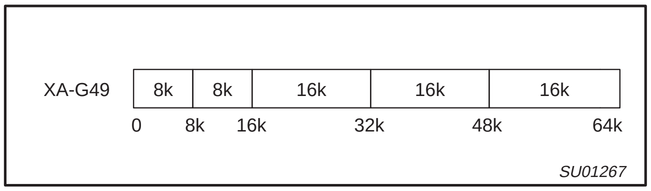

Figure 1. FLASH ROM Memory Map

The FLASH EPROM can be programmed using three different methods:

- The traditional parallel programming method
- A new In-System Programming method (ISP) through the serial port
- In Application programming method (IA) under control of a running microcontroller application program

Programming functions support the following functions:

- erase and blank check FLASH EPROM
- program and read/verify FLASH EPROM
- program and verify security bits, status byte and boot vector
- read signature bytes

IN-SYSTEM PROGRAMMING (ISP)

The In-System Programming (ISP) is performed without removing the microcontroller from the system. The In-System Programming (ISP) facility consists of a series of internal hardware resources coupled with internal firmware to facilitate remote programming of the XA-G49 through the serial port. This firmware is provided by Philips and embedded within each XA-G49 device.

The Philips In-System Programming (ISP) facility has made in-circuit programming in an embedded application possible with a minimum of additional expense in components and circuit board area.

The ISP function uses five pins: TxD0, RxD0, V_{SS}, V_{CC}, and V_{PP} (see Figure 2). Only a small connector needs to be available to interface your application to an external circuit in order to use this feature. The V_{PP} supply should be adequately decoupled and V_{PP} not allowed to exceed datasheet limits.

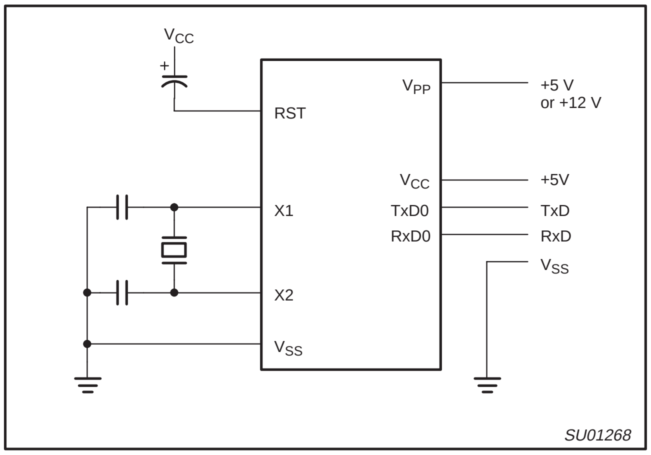

Figure 2. In-System Programming with a Minimum of Pins

Power-On Reset Code Execution

The XA-G49 contains two special FLASH registers; the BOOT VECTOR and the STATUS BYTE. At the falling edge of reset the XA-G49 examines the contents of the Status Byte. If the Status Byte is set to zero, power-up execution starts at location 0000H which is the normal start address of the user's application code. When the Status Byte is set to a value other than zero, the contents of the Boot Vector is used as the high byte of the execution address and the low byte is set to 00H. The factory default setting is 0F8H, corresponds to the address 0F800H for the factory masked-ROM ISP boot loader. A custom boot loader can be written with the Boot Vector set to the custom boot loader.

NOTE:

When erasing the Status Byte or Boot Vector, both bytes are erased at the same time. It is necessary to reprogram the Boot Vector after erasing and updating the Status Byte.

Hardware Activation of the Boot Loader

The boot loader can also be executed by holding PSEN low, EA greater than V_{IH} (such as +5 V), and ALE HIGH (or not connected) at the falling edge of RESET. This is the same effect as having a non-zero status byte. This allows an application to be built that will normally execute the end user's code but can be manually forced into ISP operation.

Programming the FLASH Registers May Require Parallel Programming

If the factory default setting for the Boot Vector (0F8H) is changed, it will no longer point to the ISP masked-ROM boot loader code. If this happens, the only way it is possible to change the contents of the Boot Vector is through the parallel programming method, provided that the end user application does not contain a customized loader that provides for erasing and reprogramming of the Boot Vector and Status Byte.

Activating In-System Programming (ISP)

The ISP feature allows programming of the FLASH EPROM through the serial port. Programming the FLASH with an executing application program is described later.

The ISP programming is accomplished by serial boot loader subroutines that are in the BOOTROM. The BOOT ROM code is located at memory address F800H and can be invoked by a selecting values for the Status Byte and the Boot Vector. After programming the FLASH, the status byte should be programmed to zero in order to allow execution of the user's application code beginning at address 0000H.

We recommend using the following sequence for ISP programming:

1. Program the Boot Vector to 0F8H to use the default serial loader.
2. Program the status byte to non-zero.
3. Perform the ISP programming.
4. Erase both Status Byte and Boot Vector after ISP has been successfully done. There is no way to erase the Status Byte without erasing the Boot Vector.
5. Program the Boot Vector back to the original value (0F8H) if the you want to keep the default serial loader as the ISP communication channel.
6. Write 00H to the Status Byte so that the program will begin at address 0000H after reset.

Using the In-System Programming (ISP)

The ISP feature allows for a wide range of baud rates to be used in your application, independent of the oscillator frequency. It is also adaptable to a wide range of oscillator frequencies. This is accomplished by measuring the bit-time of a single bit in a received character. This information is then used to program the baud rate in terms of timer counts based on the oscillator frequency. The ISP feature requires that an initial character (an lowercase f) be sent to the XA-G49 to establish the baud rate. The ISP firmware provides auto-echo of received characters.

Once baud rate initialization has been performed, the ISP firmware will only accept Intel Hex-type records. Intel Hex records consist of ASCII characters used to represent hexadecimal values and are summarized below:

```
:NNAAAARRDD..DDCC<crlf>
```

In the Intel Hex record, the “NN” represents the number of data bytes in the record. The XA-G49 will accept up to 16 (10H) data bytes per record. The “AAAA” string represents the address of the first byte in the record. If there are zero bytes in the record this field is often set to 0000. The “RR” string indicates the record type. A record type of “00” is a data record. A record type of “01” indicates the end-of-file mark. In this application additional record types will be added to indicate either commands or data for the ISP facility. The maximum number of data bytes in a record is limited to 16 (decimal). ISP commands are summarized in Table 1.

As a record is received by the XA-G49 the information in the record is stored internally and a checksum calculation is performed. The operation indicated by the record type is not performed until the entire record has been received. Should an error occur in the checksum, the XA-G49 will send an “X” out the serial port indicating a checksum error. If the checksum calculation is found to match the checksum in the record then the command will be executed. In most cases successful reception of the record will be indicated by transmitting a “.” character out the serial port (displaying the contents of the internal program memory is an exception).

In the case of a Data Record (record type 00) an additional check is made. A “.” character will NOT be sent unless the record checksum matched the calculated checksum and all of the bytes in the record were successfully programmed. For a data record an “X” indicates that the checksum failed to match and an “R” character indicates that one of the bytes did not properly program. It is necessary to send a type 02 record (specify oscillator frequency) to the XA-G49 before programming data.

The ISP facility was designed so that specific crystal frequencies were not required in order to generate baud rates or time the programming pulses. The user thus needs to provide the XA-G49 with information required to generate the proper timing. Record type 02 is provided for this purpose.

WinISP, a software utility to implement ISP programming with a PC, is available from Philips.

Table 1. Intel-Hex Records Used by In-System Programming

RECORD TYPE	COMMAND/DATA FUNCTION

00	Data Record
	`:nnaaaa00dd....ddcc`
	Where:
	`Nn        = number of bytes (hex) in record`
	`Aaaa      = memory address of first byte in record`
	`dd....dd  = data bytes`
	`cc        = checksum`
	Example:
	`:10008000AF5F67F0602703E0322CFA92007780C3FD`

01	End of File (EOF), no operation
	`:xxxxxx01cc`
	Where:
	`xxxxxx    = required field, but value is a “don’t care”`
	`cc        = checksum`
	Example:
	`:00000001FF`

02	Specify Oscillator Frequency
	`:01xxxx02ddcc`
	Where:
	`xxxx      = required field, but value is a “don’t care”`
	`dd        = integer oscillator frequency rounded down to nearest MHz`
	`cc        = checksum`
	Example:
	`:0100000210ED     (dd = 10h = 16, used for 16.0–16.9 MHz)`

03	Miscellaneous Write Functions
	`:nnxxxx03ffssddcc`
	Where:
	`nn        = number of bytes (hex) in record`
	`xxxx      = required field, but value is a “don’t care”`
	`03        = Write Function`
	`ff        = subfunction code`
	`ss        = selection code`
	`dd        = data input (as needed)`
	`cc        = checksum`
	
	Subfunction Code = 01 (Erase Blocks)
	`ff = 01`
	`ss = block number in bits 7:5, Bits 4:0 = zeros`
	Example:
	`:0200000301A05A  erase block 5`
	
	Subfunction Code = 04 (Erase Boot Vector and Status Byte)
	`ff = 04`
	`ss = don’t care`
	`dd = don’t care`
	Example:
	`:0100000304F8  erase boot vector and status byte`
	
	Subfunction Code = 05 (Program Security Bits)
	`ff = 05`
	`ss = 00 program security bit 1`
	`     01 program security bit 2`
	`     02 program security bit 3`
	Example:
	`:020000030501F5  program security bit 2`
	
	Subfunction Code = 06 (Program Status Byte or Boot Vector)
	`ff = 06`
	`ss = 00 program status byte`
	`     01 program boot vector`
	Example:
	`:020000030601F4  program boot vector`
	`:020000030601F8FC to F8`

RECORD TYPE	COMMAND/DATA FUNCTION

04	Display Device Data or Blank Check – Record type 04 causes the contents of the entire FLASH array to be sent out the serial port in a formatted display. This display consists of an address and the contents of 16 bytes starting with that address. No display of the device contents will occur if security bit 2 has been programmed. The dumping of the device data to the serial port is terminated by the reception of any character.
	
	General Format of Function 04
	`:05xxxx04sssseeeeffcc`
	Where:
	`05   = number of bytes (hex) in record`
	`xxxx = required field, but value is a “don’t care”`
	`04   = “Display Device Data or Blank Check” function code`
	`ssss = starting address`
	`eeee = ending address`
	`ff   = subfunction`
	`       00 = display data`
	`       01 = blank check`
	`cc   = checksum`
	Example:
	`:0500000440004FFF0069  display 4000–4FFF`

05	Miscellaneous Read Functions
	
	General Format of Function 05
	`:02xxxx05ffsscc`
	Where:
	`02   = number of bytes (hex) in record`
	`xxxx = required field, but value is a “don’t care”`
	`05   = “Miscellaneous Read” function code`
	`ffss = subfunction and selection code`
	`       0000 = read signature byte – manufacturer id (15H)`
	`       0001 = read signature byte – device id # 1   (EAH)`
	`       0002 = read signature byte – device id # 2`
	`                                         (XA-G49 = 54H)`
	`       0700 = read security bits`
	`       0701 = read status byte`
	`       0702 = read boot vector`
	`cc   = checksum`
	Example:
	`:020000050001F8  read signature byte – device id # 1`

WINISP – The Windows In-System Programmer Utility Program

The WINISP program is available from your Philips representative, or from the Philips website.

Attaching the ISP device to a computer

The device is connected to a computer using a DB9/9 or DB9/25 RS-232 serial interface cable. The 9 pin male connector plugs into the rear serial socket of the device while the other end (9 or 25 pin female connector, depending on the computers attached serial port) plugging into an available serial port in the rear of the computer.

Power to the unit can be provided by means of 7.5 V to 12 V power adapter.

Installing the microcontroller into the ISP BOARD

With the controller socket in the down position, place the chip in the socket with the chip key matching the socket key. Gently push the chip down into the socket until the chip is completely inserted.

ISP Switches and buttons

The ISP contains a RESET button and a NORMAL/TEST jumper. These switches are used in combination during the programming and erasing of FLASH sectors.

Configuring the ISP Program

Launch the ISP program into a window. Use the mouse to select the part type, the Windows serial port being used, and the oscillator frequency.

CHIP – selects the chip type:

- XA-G49

PORT – Selects which port on the host computer is connected to the ISP board

- COM1
- COM2

RANGE – Selects the beginning and ending address

- START
- END

ISP Programmer System Commands

Load File
Click the LOAD FILE button and enter the desired file name into the dialog box

Erase Blocks
Click the ERASE BLOCKS button and use the mouse to select the desired blocks. Click the ERASE! button, then press the RESET button on the ISP board

Blank Check
Click the BLANK CHECK button. Press the RESET button on the ISP board

Program Part
Click the PROGRAM PART button. Press the RESET button on the ISP board

Read Part
Click the READ PART button. Press the RESET button on the ISP board

Verify Part
Click the VERIFY PART button. Press the RESET button on the ISP board

Fill Buffer
Enter the starting and ending address in the RANGE boxes. Click the FILL BUFFER button. Enter the data pattern in the next dialog box.

NOTE: The XA-G49 must be running the BOOT ROM ISP program for the Windows ISP to be able to communicate with the microcontroller. The TEST jumper is connected to the PSEN line on the microcontroller. Set the TEST switch to “TEST” (ground) and then press and release the RESET switch to start the ISP serial boot loader. Return the TEST switch to “NORMAL” (floating) and press RESET to start the user’s application program.

In-Application Programming Method

Several Application Program Interface (API) calls are available for use by an application program to permit selective erasing and programming of FLASH sectors. All calls are made through a common interface, PGM_MTP. The programming functions are selected by setting up the microcontroller's registers before making a call to PGM_MTP at FFF0H. The oscillator frequency is an integer number rounded down to the nearest megahertz. For example, set R0 to 22 for 22.1184 MHz. Results are returned in the registers. The API calls are shown in Table 2.

Table 2. API calls

API CALL	PARAMETER

PROGRAM DATA BYTE	Input Parameters:
	`ROH = 02h`
	`R6 = address of byte to program`
	`R4L = byte to program`
	Return Parameter
	`R4L = 00 if pass, !00 if fail`

ERASE BLOCK	Input Parameters:
	`R0H = 01h`
	`R6 = block number in bits 7:5, bits 4:0 = ’0’`
	Return Parameter
	`R4L = 00 if pass, !00 if fail`

ERASE BOOT VECTOR	Input Parameters:
	`R0H = 04h`
	`R6 = don’t care`
	Return Parameter
	`R4L = 00 if pass, !00 if fail`

PROGRAM SECURITY BIT	Input Parameters:
	`R0H = 05h`
	`R6 = 0000h – security bit # 1 (inhibit writing to FLASH)`
	`00001h – security bit # 2 (inhibit FLASH verify)`
	`00002h – security bit # 3 (disable external memory)`
	Return Parameter
	`none`

PROGRAM STATUS BYTE	Input Parameters:
	`R0H = 06h`
	`R6 = 0000h – program status byte`
	`R4L = status byte`
	Return Parameter
	`R4L = 00 if pass, !00 if fail`

PROGRAM BOOT VECTOR	Input Parameters:
	`R0H = 06h`
	`R6 = 0001h – program boot vector`
	`R4L = boot vector`
	Return Parameter
	`R4L = 00 if pass, !00 if fail`

READ DEVICE DATA	Input Parameters:
	`R1 = 03h`
	`R6 = address of byte to read`
	Return Parameter
	`R4L = value of byte read`

READ MANUFACTURER ID	Input Parameters:
	`R0H = 00h`
	`R6 = 0000h (manufacturer ID)`
	Return Parameter
	`R4L = value of byte read`

READ DEVICE ID # 1	Input Parameters:
	`R0H = 00h`
	`R6 = 0001h (device ID # 1)`
	Return Parameter
	`R4L = value of byte read`

READ DEVICE ID # 2	Input Parameters:
	`R0H = 00h`
	`R6 = 0002h (device ID # 2)`
	Return Parameter
	`R4L = value of byte read`

API CALL	PARAMETER

READ SECURITY BITS	Input Parameters:
	`R0H = 07h`
	`R6 = 0000h (security bits)`
	Return Parameter
	`R4L = value of byte read`

READ STATUS BYTE	Input Parameters:
	`R0H = 07h`
	`R6 = 0001h (status byte)`
	Return Parameter
	`R4L = value of byte read`

READ BOOT VECTOR	Input Parameters:
	`R0H = 07h`
	`R6 = 0002h (boot vector)`
	Return Parameter
	`R4L = value of byte read`

APPENDIX A — Installing the WINISP Programmer Software (Win95 / Win98 / WinNT / Win3.1)

Package Contents:

There are two disks included with the installation package labeled ‘SETUP DISK 1’ and ‘SETUP DISK 2’.

Minimum System Requirements:

33 MHz, 386 processor

8 MB RAM

Win95, Win98, WinNT

Installation:

Close all open programs before beginning installation.

Insert disk labeled ‘SETUP DISK 1’ into drive A:.

From the Start menu select Run...

Type ‘A:\Setup.exe’, click OK

Setup cannot install system files or update shared files if they are in use. Before proceeding, we recommend that you close any applications that you may be running.

Make sure all other open programs are closed, click OK.

WINISP Install defaults to C:\WINISP. Click on the large setup-icon button to continue or the ‘CHANGE DIRECTORY’ button to change the install directory.

Philips Semiconductors
811 East Arques Avenue
P.O. Box 3409
Sunnyvale, California 94088–3409
Telephone 800-234-7381

Date of release: 09-99

Document order number: 9397 750 06453

Let's make things better.

Philips
Semiconductors

PHILIPS