APPLICATION NOTE

ABSTRACT
This application note provides code samples in C and assembly, which will help the end user use the In-Application Programming (IAP) technique for programming the on-chip Flash.

AN10256
Philips LPC210x microcontroller family

Amitkumar Bhojraj

2003 Dec 12

Philips Semiconductors

PHILIPS

INTRODUCTION

In–Application (IAP) programming is performing erase and write operations on the on–chip Flash memory as directed by the end–user application code. The Flash boot loader provides the interface for programming the Flash memory. For detailed information on the In–Application Programming please refer to the Flash Memory System and Programming chapter in the LPC210x User Manual. In this application note, code samples are provided in C and assembly, which show how IAP may be used. The IAP routine resides at 0x7FFFFFF0 and is Thumb code.

IAP CODE IN C

The IAP function could be called in the following way using C. This section is taken from the User Manual.

Define the IAP location entry point. Since the 0th bit of the IAP location is set there will be a change to the Thumb instruction set when the program counter branches to this address.

```
#define IAP_LOCATION 0x7ffffff1
```

Define data structure or pointers to pass IAP command table and result table to the IAP function

```
unsigned long command[5];
unsigned long result[2];
```

or

```
unsigned long * command;
unsigned long * result;
command=(unsigned long *) 0x…..
result= (unsigned long *) 0x…..
```

Define pointer to function type, which takes two parameters and returns void. Note the IAP returns the result with the base address of the table residing in R1.

```
typedef void (*IAP)(unsigned int [],unsigned int[]);
IAP iap_entry;
```

Setting function pointer

```
iap_entry=(IAP) IAP_LOCATION;
```

Whenever user wishes to call IAP within the application, the following statement could be used.

```
iap_entry (command, result);
```

IAP CODE IN ASSEMBLY

The IAP routine may be called in the following way using ARM assembly code. This code was developed using the ARM Developer Suite (ADS1.2). The assembler directives will change depending upon the assembler the end–user will use.

```
;--------------------------------------------------------
              AREA arm_code, CODE
              CODE32
              EXPORT initial        ; This routine could be
                                    ; linked to other
                                    ; routines using this
                                    ; global symbol

;--------------------------------------------------------
;             Symbol definitions
;--------------------------------------------------------

IAP_ENTRY     EQU 0x7ffffff1        ; IAP entry point
COMMAND       EQU 0x…..             ; Command table pointer
RESULT        EQU 0x…..             ; Result table pointer
;--------------------------------------------------------
;              Main
;--------------------------------------------------------

initial
              STMFD SP!,{R0–R2,R14} ; Push the register set
                                    ; and link register into
                                    ; stack
```

```
                LDR R0,=COMMAND       ; Set the pointers for
                LDR R1,=RESULT        ; command and result
                                      ; tables

;-------------------------------------------------------
; Once the pointers are set, the command code and its
; respective parameters need to be stored in the command
; table. An example is provided below where the command
; code (54) for IAP command “Read Part ID” is stored into
; the command table
;-------------------------------------------------------

                MOV R2,#0x36
                STR R2, [R0]

;-------------------------------------------------------
; Please look below (after END) for description for how the
; IAP routine is called
;-------------------------------------------------------

                BL jump_to_IAP

;-------------------------------------------------------
; At this point user has to analyze the result table and
; take action depending upon the status code returned by
; the IAP routine. (Code not shown)
;-------------------------------------------------------

                LDMFD SP!,{R0–R2,R14} ; Pop link register
                                      ; and register workspace
                MOV PC,LR

;-------------------------------------------------------
;               Call IAP routine
;-------------------------------------------------------

jump_to_IAP
                LDR R12,=IAP_ENTRY
                BX R12                ; Branch to 0x7FFFFFF1
                                      ; and Change to thumb
                                      ; instruction set
                END
```

To call the IAP function, we branch and link (BL) to a small routine jump_to_IAP and then we call the IAP function using BX. By performing BL jump_to_IAP we get R14 to point to the next instruction and then using BX instruction we can directly jump to the IAP routine and change to Thumb instruction set.

If user wishes to call the IAP routine using Thumb code, then the code could be as follows.

```
;-------------------------------------------------------
                AREA thumb_code, CODE
                CODE16
                EXPORT initial        ; this routine could be
                                      ; linked to other
                                      ; routines using this
                                      ; global symbol
;-------------------------------------------------------
;               Symbol definitions
;-------------------------------------------------------

IAP_ENTRY       EQU 0x7ffffff1        ; IAP entry point
COMMAND         EQU 0x……              ; Command table pointer
RESULT          EQU 0x……              ; Result table pointer
;-------------------------------------------------------
;                Main
;-------------------------------------------------------

Initial
                PUSH {R0–R2,R14}      ; Push the register
                                      ; workspace and link
```

```
                                ; register into stack
              LDR R0,=COMMAND   ; Set the pointers for
              LDR R1,=RESULT    ; command and
                                ; result tables
;-------------------------------------------------------
; Once the pointers are set, the command code and its
; respective parameters need to be stored in the command
; table. An example is provided below where the command
; code (54) for IAP command “Read Part ID” is stored into
; the command table
;-------------------------------------------------------
              MOV R2,#0x36
              STR R2, [R0]
;-------------------------------------------------------
; Please look below (after END) for description for how the
; IAP routine is called
;-------------------------------------------------------
              BL jump_to_IAP
;-------------------------------------------------------
; At this point user has to analyze the result table and
; take action depending upon the status code returned by
; the IAP routine. (Code not shown)
;-------------------------------------------------------
              POP{R0–R2,R3}
              BX R3             ; Pop the link register
                                ; contents and go back to
                                ; ARM mode
;-------------------------------------------------------
;             Call IAP routine
;-------------------------------------------------------
jump_to_IAP
              LDR R2,=IAP_ENTRY
              BX R2
              END
```

The differences in the Thumb code as compared to the ARM code being the assembler directive CODE16 and the push and pop instructions for the stack.

USING THE ARM DEVELOPER SUITE (ADS 1.2) TOOLS

There is one more way of calling the IAP routine using the *symbol definitions (symdefs) file* but this is specific to the ARM development tools. The IAP routine could be looked as an image residing in Flash. Now, an image residing in RAM can access the global symbols of this image residing in Flash using the symdefs file. The symdefs file can be considered to be an object file, which contains symbols and their values. Please refer to Chapter 4 in the ARM Developer Suite Linker and Utilities Guide for detailed information on accessing symbols.

The symdefs file could be defined as follows for the IAP routine.

```
#<SYMDEFS># ARM Linker, ADS1.2 [Build 805]:Last Updated: Fri Jun 06 15:46:24 2003
0x7ffffff0 T iap_entry
```

The first 11 characters #<SYMDEFS># of this text file recognizes this file as a symdefs file. We then provide the symbol information with regard to the IAP routine in the second line. This file could then be linked to user application using the -F option at command line for the ARM linker. Please click on Project Settings, then on ARM linker. To do this on the Metrowerks CodeWarrior, open the Debug settings window for the project, then click on ARM linker and then “Equivalent Command Line” could be seen (under Output tab) where the following option could be added:

```
–F C:\...\symdefs
```

where **symdefs** is the symdefs file.

Once the symdefs file has been defined and added the to the project using the -F option, then in the user application the following needs to be done (Only C code is shown as an example):

Define data structure or pointers for IAP command table and result table

```
unsigned long command[5];
unsigned long result[2];
```

or

```
unsigned long * command;
unsigned long * result;
command=(unsigned long *) 0x…...
result= (unsigned long *) 0x…...
```

Call IAP routine

```
iap_entry(command,result);
```

As seen above, iap_entry does not have to be defined anywhere in the application, as the linker now knows it is been defined in the image residing in Flash through the symdefs file.

Definitions

Short-form specification — The data in a short-form specification is extracted from a full data sheet with the same type number and title. For detailed information see the relevant data sheet or data handbook.

Limiting values definition — Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 60134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.

Application information — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Disclaimers

Life support — These products are not designed for use in life support appliances, devices or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

Right to make changes — Philips Semiconductors reserves the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

Contact information

For additional information please visit
http://www.semiconductors.philips.com. Fax: **+31 40 27 24825**

For sales offices addresses send e-mail to:
sales.addresses@www.semiconductors.philips.com.

Date of release: 12-03

Document order number: 9397 750 12483

Let's make things better.

Philips Semiconductors

PHILIPS