

---

# I.MX Platform Advanced Toolkit Reference Manual

Standard Version

Version 1.66  
1/2009



**How to Reach Us:**

**Home Page:**

www.freescale.com

**E-mail:**

support@freescale.com

**USA/Europe or Locations Not Listed:**

Freescale Semiconductor  
Technical Information Center, CH370  
1300 N. Alma School Road  
Chandler, Arizona 85224  
+1-800-521-6274 or +1-480-768-2130  
support@freescale.com

**Europe, Middle East, and Africa:**

Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
support@freescale.com

**Japan:**

Freescale Semiconductor Japan Ltd.  
Headquarters  
ARCO Tower 15F  
1-8-1, Shimo-Meguro, Meguro-ku,  
Tokyo 153-0064, Japan  
0120 191014 or +81 3 5437 9125  
support.japan@freescale.com

**Asia/Pacific:**

Freescale Semiconductor China Ltd.  
Exchange Building 23F, No. 118 Jianguo Road  
Chaoyang District  
Beijing 100022, China  
+86 010 5879 8000  
support.asia@freescale.com

**For Literature Requests Only:**

Freescale Semiconductor Literature Distribution Center  
P.O. Box 5405  
Denver, Colorado 80217  
1-800-521-6274 or 303-675-2140  
Fax: 303-675-2150  
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale and the Freescale logo are trademarks or registered trademarks of Freescale Semiconductor, Inc. in the U.S. and other countries. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2008. All rights reserved.



# Contents

## Chapter 1 Introduction

1.1	ATK Components and Environment .....	1-1
1.2	Source Code Tree .....	1-2

## Chapter 2 Build Procedure

2.1	Set Up the Build Environment .....	2-1
2.1.1	Install Source Code Tree .....	2-1
2.1.2	Get/Install Development/Build Tools .....	2-1
2.2	Build Device Program with Cygwin/armgcc .....	2-2
2.3	Building the Host DLL and GUI Application .....	2-4
2.3.1	Build Host DLL .....	2-5
2.3.2	Build the GUI Application .....	2-5
2.4	Generate the ATK Package .....	2-6

## Chapter 3 Development Guidelines

3.1	APIs of Device Program .....	3-1
3.1.1	Device RAM Kernel .....	3-1
3.1.1.1	RAM Kernel Protocol .....	3-1
3.1.1.2	Common Commands/Responses .....	3-2
3.1.1.2.1	Reset .....	3-2
3.1.1.2.2	Download .....	3-2
3.1.1.2.3	Execute .....	3-2
3.1.1.2.4	Get version .....	3-2
3.1.1.3	Flash Commands/Responses: .....	3-3
3.1.1.3.1	Flash initialize .....	3-3
3.1.1.3.2	Flash erase .....	3-3
3.1.1.3.3	Flash dump .....	3-3
3.1.1.3.4	Flash program (page or block program) .....	3-3
3.1.1.3.5	Flash program (un-boundary program) .....	3-3
3.1.1.3.6	Flash get capacity .....	3-4
3.1.1.4	Extend Command/Response: .....	3-4
3.1.1.4.1	Switch from UART to USB connection .....	3-4
3.1.1.4.2	Swap BI SWAP Flag .....	3-4
3.1.1.4.3	Set BBT Flag .....	3-4
3.1.1.4.4	Set interleave Flag .....	3-4
3.1.1.5	Source Code of Device RAM Kernel .....	3-4
3.1.2	Flash Library .....	3-5
3.1.2.1	Device Flash Interfaces .....	3-5
3.1.2.2	Source Code of Device Flash Library .....	3-7
3.1.3	GNU Building Environment for Device Program .....	3-7

3.2	APIs of ROM/RAM Kernel Host DLL.....	3-8
3.2.1	Interfaces for ROM Kernel.....	3-8
3.2.2	Interfaces for Host RAM Kernel.....	3-8
3.2.3	Common APIs for Host DLL.....	3-10
3.3	GUI Application—Use Cases.....	3-10
3.3.1	UART Use Case.....	3-11
3.3.2	USB Use Case.....	3-12
3.3.3	Flash Programming Use Case.....	3-13

## Chapter 4 Additional Information

4.1	Supporting a New Flash Device.....	4-1
4.1.1	NAND Flash.....	4-1
4.1.2	NOR Flash.....	4-2
4.1.2.1	Generate new SPANSION NOR Flash Model.....	4-2
4.1.2.2	Support New Flash Model.....	4-2
4.2	Developing a Device Program using CodeWarrior.....	4-3
4.3	Why the Flash Operations need Response Callback.....	4-3
4.4	Installing Cygwin for the ATK.....	4-4

1-1	ATK Components and Environment .....	1-1
3-1	UART Use Case .....	3-11
3-2	USB Use Case .....	3-12
3-3	Flash Programming Use Case .....	3-13
4-1	Installation Setup.....	4-4
4-2	Download Screen .....	4-5
4-3	Installation Directory Screen .....	4-6
4-4	Local Directory Screen .....	4-7
4-5	Select Package Screen.....	4-8
4-6	make .....	4-9
4-7	Installation.....	4-9
4-8	Creating Icons .....	4-10
4-9	Installed.....	4-10



1-1	Recommended Source Code Tree Structure for ATK.....	1-2
2-1	Where to Obtain Build/Development Tools.....	2-1
2-2	Build Commands.....	2-2
2-3	Example Build Commands for building Flash and Dummy Fuse Libraries.....	2-3
2-4	Copy the Host DLL Files .....	2-5
3-1	Command/Response Package Format .....	3-1
3-2	RAM Kernel Protocol, Compact Format .....	3-1
3-3	Source files for Device RAM Kernel.....	3-4
3-4	APIs of the Device Flash Library .....	3-5
3-5	Source Files of Device Flash Library .....	3-7
3-6	Source Files of GNU Environment for Device Program .....	3-7
3-7	APIs of Host ROM Kernel DLL .....	3-8
3-8	APIs of Host RAM Kernel DLL .....	3-9
3-9	Common APIs of the Host DLL .....	3-10
4-1	Source Files for the Device RAM Kernel.....	4-3





# About This Book

This guide describes the integration environment of the Advanced Toolkit (ATK), including the source code tree, build process, and interfaces of the software components.

## Audience

This document is intended for software, hardware, and system engineers who are planning to use the product and for anyone who wants to understand more about the product.

## Conventions

This document uses the following conventions:

- Courier* Is used to identify commands, explicit command parameters, code examples, expressions, data types, and directives.
- Italic* Is used for emphasis, to identify new terms, and for replaceable command parameters.

## Reference

The following documents were referenced to build this document.

1. i.MX31 ROM User's Guide

## Organization

This document contains the following chapters.

- [Chapter 1, "Introduction,"](#) describes the ATK components and source code tree.
- [Chapter 2, "Build Procedure,"](#) explains how to build the ATK source codes.
- [Chapter 3, "Development Guidelines,"](#) introduces the APIs for each component.
- [Chapter 4, "Additional Information,"](#) responds to frequently asked questions.

## Definitions, Acronyms, and Abbreviations

[Table i](#) contains acronyms and abbreviations used in this document.

**Table i. Acronyms and Abbreviations**

Term	Definition
ATK	Advanced Toolkit
RKL	Ram Kernel

**Table i. Acronyms and Abbreviations (continued)**

<b>Term</b>	<b>Definition</b>
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

# Chapter 1

## Introduction

The standard version of the Advanced Toolkit (ATK) is a container of tools for the i.MX processor family. It integrates two main tools:

- Flash tool for programming/dumping/erasing
- Utility image conversion tool
- 

You can use the ATK to develop applications for the i.MX processors. As it is, the ATK is not suitable to be used as a mass production tool. However, using this document as reference, you can develop a mass production tool.

### 1.1 ATK Components and Environment

Figure 1-1 illustrates the ATK components and environment in which the ATK is used.

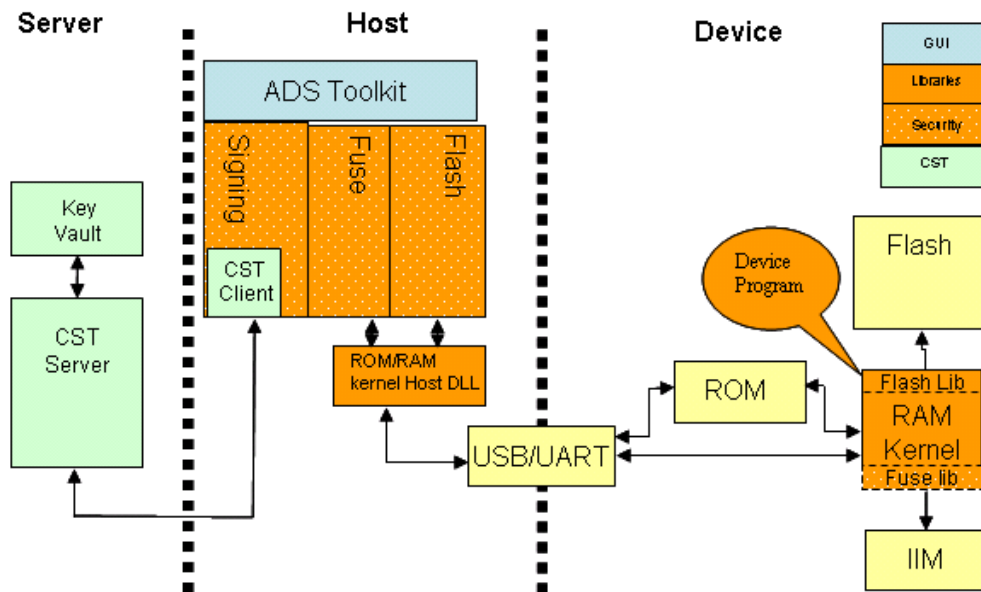


Figure 1-1. ATK Components and Environment

The PC host communicates with the target board (containing an i.MX processor) over a USB or UART link.

#### NOTE

The fuse and code signing functionalities are reserved in the standard ATK package.

The ATK provides the following components:

1. A GUI application that supports the following tools:
  - A Flash Tool that enables you to program/dump/erase Flash memory on the target board.
  - An image conversion tool.
2. A Host DLL (Dynamic Link Library), which supports ROM bootstrap and RAM kernel protocols. The Host DLL sends commands to the device and receives responses from the PC host through the USB/UART. The Host DLL is used by the Flash and fuse functions.
3. A device program that runs in the target board RAM and executes operations specified by the hardware:
  - A RAM kernel library, which supports RAM kernel protocols between the host and device (handling commands from the PC host and sending responses to the device).
  - A Flash library, which provides the interfaces to the Flash media, such as NOR Flash, NAND Flash, MMC, and others.

## 1.2 Source Code Tree

In order to make ATK integration support straightforward, we recommend that you structure your source code tree to be similar to that listed in [Table 1-1](#).

**Table 1-1. Recommended Source Code Tree Structure for ATK**

Directory	Subdirectory	Description	
device_program/	—	Device program, which includes RAM kernel, Flash, and fuse libraries <b>Note:</b> The source codes for fuse library are obtainable only under NDA. Contact your local Freescale support for details.	
	bin/	Location for storing of output binary files, map files, and elf files after building	
	cw_mcp/	Examples for Code Warrior projects; these files are references for developers who will use Code Warrior as the development tool for device programs.	
	Flash/	Source code for different Flash libraries.	
	init/	Initialization code for GNU environment.	
	ram_kernel/	RAM kernel library	
	global_inc/	Global header files	
gui_application/		ATK GUI application	
	bin	DLL and executable programs when running CST and image conversion tools	
	Config	Configuration files used when running Flash	
	GridCtrl_src	Code for the grid needed by the DCD tool.	
	Image	Device program libraries with RAM kernel, Flash for i.MX processors.	
	Platform		Code that is dependent on the hardware platform.
			Host DLL to support ROM bootstrap and RAM kernel protocols.

**Table 1-1. Recommended Source Code Tree Structure for ATK (continued)**

<b>Directory</b>	<b>Subdirectory</b>	<b>Description</b>
host_dll/	ComPort	Code for serial port operations.
	MXUsb.lib	USB library, which was developed using the Jungo development kit.



## Chapter 2

# Build Procedure

This chapter explains how to build the ATK source codes.

## 2.1 Set Up the Build Environment

### 2.1.1 Install Source Code Tree

Obtain the ATK source code package and run the install execution `FSL_ATK_SOURCE_CODE_STD_<version_number>.exe` to install the source code to a directory `<ATK_SRC_PATH>` (for example, `atk_src`) on a PC with Windows XP/Windows 2000 OS.

#### NOTE

Ensure that the layout of your source code tree under the `<ATK_SRC_PATH>` is the same as that shown in [Table 2-1](#).

### 2.1.2 Get/Install Development/Build Tools

To build the host DLL and guide applications, Microsoft Visual C++ 2005 must be installed.

To build the device program, Cygwin and GNU gcc ([Table 2-1](#)) must be installed.

**Table 2-1. Where to Obtain Build/Development Tools**

Software	Download from
Cygwin	from <a href="http://www.cygwin.com/">http://www.cygwin.com/</a> . For Cygwin installation instructions, see section 4.5. Installing Cygwin for ATK.
GNU gcc toolchain for cygwin	4.1.1 version under <a href="http://www.gnuarm.com/bu-2.17_gcc-4.1.1-c-c++_nl-1.14.0_gi-6.5.exe">http://www.gnuarm.com/bu-2.17_gcc-4.1.1-c-c++_nl-1.14.0_gi-6.5.exe</a> To install GNU gcc, follow the installation instructions and use the default settings.

#### NOTE

Make sure to select the correct `cygwin1.dll` for building. This `cygwin1.dll` for the ARM-ELF tool chain is located under the GNUARM bin directory after GNU gcc is installed. For example:

```
C:\Program Files\GNUARM\bin
```

Once some issues happen, which should be caused by the collisions of cygwin1.dlls or other cygwin dlls, respectively, in C:\Program Files\GNUARM\bin and C:\cygwin\bin, you must copy all cygwin.dll files under C:\Program Files\GNUARM\bin to C:\cygwin\bin or any directory where Cygwin installed.

## 2.2 Build Device Program with Cygwin/armgcc

This section describes the tools and commands you will need.

1. Ensure that the tools listed in [Section 2.1.2, “Get/Install Development/Build Tools,”](#) are installed.
2. Launch cygwin Bash shell.
3. Change directory `cd <ATK_SRC_PATH>/device_program/`  
e.g. `$cd /cygdrive/d/atk_src/device_program/`
4. Enter the build commands according to build targets ([Table 2-2](#)).

**Table 2-2. Build Commands**

Build Commands	Comments
make clean	Clean all unnecessary output files (such as object files) when generating during the build procedure.
make MCU={user input} REV={user input} flashlib FLASH_TYPE={user input} FLASH_MODEL={user input}	Build the device program with MCU type, Revision Version, Flash type, and model .The option "flashlib" will build only the Flash library and the fuse library will be built out. For command details, see <a href="#">Table 2-3 (b)</a> . <b>Note:</b> If you have only the standard source codes package, use this build command.
make MCU={user input} unittest	Build device program with dummy Flash and fuse library.

Optional compiler flags for IC\_TYPE and FLASH\_TYPE:

- MCU: indicates the type of IC.
  - mx31: i.MX31 chip
  - mx32: i.MX32 chip
  - mx27: i.MX27 chip
  - mx35\_: i.MX35 chip
  - mx37: i.MX37 chip
  - mx51: i.MX51 chip
  - mx25: i.MX25 chip
- REV: indicate Revision version of the chip specified by MCU. it is used to distinguish the different images for different revisions of the same chip type. Currently, it is used to i.MX51 TO2 NAND, i.MX35 TO1/TO2 MMC/SD and i.MX51 TO1/TO2 MMC/SD
  - to1: Revision version 2
  - to2: Revision version 2
- FLASH\_TYPE: indicates the type of Flash.



- nor: NOR Flash
- nand: NAND Flash
- mmc: MMC
- sd: SD
- FLASH\_MODEL: indicates the Flash model
  - spansion: NOR Flash S71WS256ND0/SG29GL512N

### NOTE

From Version 1.5.1, FLASH\_MODEL should be obsolete if FLASH\_TYPE=nand.

For more details about build flags, see Makefile under: <ATK\_SRC\_PATH>device\_program/Flash/

i.MX27/i.MX31/i.MX32/i.MX35/i.MX37/i.MX51/i.MX25 Table 2-3 provides example commands for building Flash and dummy fuse libraries for the i.MX27/i.MX31/i.MX32/i.MX35/i.MX37/i.MX51/i.MX25 boards.

**Table 2-3. Example Build Commands for building Flash and Dummy Fuse Libraries**

Build a device program for	Featuring	Command
MGN board and 3-stack board	i.MX31 chip and K9F4G08U0M/ K9K2G08R0 NAND Flash	make MCU=mx31 flashlib FLASH_TYPE=nand
i.MX31ADS board	i.MX31 chip, K9K1G08U0B NAND Flash	make MCU=mx31 flashlib FLASH_TYPE=nand
	i.MX31 chip, MMC card	make MCU=mx31 flashlib FLASH_TYPE=nand
	i.MX31 chip, S71WS256ND0 NOR Flash	make MCU=mx31 flashlib FLASH_TYPE=nor FLASH_MODEL=spansion
i.MX32ADS board	i.MX32 chip, K9F4G08U0M/ K9K2G08R0 NAND Flash	make MCU=mx32 flashlib FLASH_TYPE=nand
	i.MX32 chip, K9K1G08U0B NAND Flash	make MCU=mx32 flashlib FLASH_TYPE=nand
	i.MX32 chip, MMC card	make MCU=mx32 flashlib FLASH_TYPE=mmc
	i.MX32 chip, S71WS256ND0 NOR Flash	make MCU=mx32 flashlib FLASH_TYPE=nor FLASH_MODEL=spansion
i.MX27ADS board	i.MX27 chip, K9K1G08U0B NAND Flash	make MCU=mx27 flashlib FLASH_TYPE=nand
	i.MX27 chip, S71WS256ND0 NOR Flash	make MCU=mx27 flashlib FLASH_TYPE=nor FLASH_MODEL=spansion
i.MX27 3-Stack HW	i.MX27 TO2 chip, K9K2G08R0 NAND Flash	make MCU=mx27 flashlib FLASH_TYPE=nand

Table 2-3. Example Build Commands for building Flash and Dummy Fuse Libraries (continued)

Build a device program for	Featuring	Command
i.MX35 TO1/TO2 3-Stack HW	i.MX35 chip, K9LAG08U0M NAND Flash	make MCU=mx35 flashlib FLASH_TYPE=nand
	i.MX35 chip, SG29GL512N NOR Flash	make MCU=mx35 flashlib FLASH_TYPE=nor FLASH_MODEL=spanion
	i.MX35 TO1 chip , MMC/SD card	make MCU=mx35 REV=to1 flashlib FLASH_TYPE=mmc
	i.MX35 TO2 chip ,MMC/SD card	make MCU=mx35 REV=to2 flashlib FLASH_TYPE=mmc
i.MX37 3-Stack HW	i.MX37 chip, K9LBG08U0M NAND Flash	make MCU=mx37 flashlib FLASH_TYPE=nand
	i.MX37 chip, MMC card	make MCU=mx37 flashlib FLASH_TYPE=mmc
i.MX51 TO1 3-Stack HW	i.MX51 TO1 chip, K9LBG08U0M NAND Flash	make MCU=mx51 flashlib FLASH_TYPE=nand
	i.MX51 TO1 chip, MMC/SD card	make MCU=mx51 flashlib FLASH_TYPE=mmc
i.MX51 TO2 3-Stack HW	i.MX51 TO2 chip, MT29F32G08QAA NAND Flash	make MCU=mx51 REV=to2 flashlib FLASH_TYPE=nand
	i.MX51 TO2 chip, MMC/SD card	make MCU=mx51 REV=to2 flashlib FLASH_TYPE=mmc
i.MX25 3-Stack HW	i.MX25 chip, K9LAG08U0M NAND Flash	make MCU=mx25 flashlib FLASH_TYPE=nand
	i.MX25 chip, MMC/SD card	make MCU=mx25 flashlib FLASH_TYPE=mmc

**NOTE**

Execute “make clean” between different build commands.

5. Install the generated files in <ATK\_SRC\_PATH>/device\_program/bin to the <ATK\_SRC\_PATH>/gui\_application/image folder using the command:

```
make install DEST=../gui_application/image/
```

## 2.3 Building the Host DLL and GUI Application

In the ATK integration environment, the host dll and GUI applications are developed and built using Visual C++ 2005.

## 2.3.1 Build Host DLL

To build the host DLL, use these steps:

1. Open `<ATK_SRC_PATH>\host_dll\AtkHostApi_std.dsw` and set **Build > Configuration Manager** as **Win32 Release** for a release version or as **Win32 Debug** for a debug version.
2. Check **Project > Properties > Configuration Properties > General > Project Defaults > Use of MFC** as **Use MFC in a Static Library**.
3. Check **Project > Properties > Configuration Properties > Linker > Input > Additional Dependencies** to make sure `MXUsb.lib` is linked.
4. Select **Build > Rebuild AtkHostApi**. After building is complete, install the generated files under `<ATK_SRC_PATH>\host_dll\Release\` to `<ATK_SRC_PATH>\gui_application\Release`  
**OR**  
`<ATK_SRC_PATH>\host_dll\Debug\` to `<ATK_SRC_PATH>\gui_application\Debug`

Table 2-4 provides the copy locations for the host DLL files.

Table 2-4. Copy the Host DLL Files

Copy from	To
<code>&lt;ATK_SRC_PATH&gt;\host_dll\Release\AtkHostApi_std.dll</code>	<code>&lt;ATK_SRC_PATH&gt;\gui_application\Release</code>
<code>&lt;ATK_SRC_PATH&gt;\host_dll\Debug\AtkHostApi_std.dll</code>	<code>&lt;ATK_SRC_PATH&gt;\gui_application\Debug</code>

## 2.3.2 Build the GUI Application

To build the GUI application, use these steps:

1. Open `<ATK_SRC_PATH>\gui_application\ADSToolkit_std.dsw`.
2. Set **Build > Configuration Manager** as **Win32 Release** for a release version or as **Win32 Debug** for a debug version.
3. Set **Project > Properties > Configuration Properties > General > Project Defaults > Use of MFC** as **Use MFC in a Static Library**.
4. Check **Project > Properties > Configuration Properties > Linker > Input > Additional Dependencies** to make sure that `AtkHostApi_std.lib` is linked to the right location. By default `AtkHostApi_std.lib` is linked to one of the following:  
`../host_dll/Release/AtkHostApi_std.lib`  
**OR**  
`../host_dll/Debug/AtkHostApi_std.lib`
5. Select **Build > Rebuild ADSToolkit**.  
The output file `ADSToolkit_std.exe` is located in the one of the following `<ATK_SRC_PATH>\gui_application\Release`, **OR**  
`<ATK_SRC_PATH>\gui_application\Debug` directory.

## 2.4 Generate the ATK Package

To run ATK normally, generate the ATK package:

1. If `ADSToolkit.exe` is located under package location `<ATK_TOOL_PATH>`. Then install `AtkHostApi_std.dll` to `<ATK_TOOL_PATH>`.
2. Copy the following directories to `<ATK_TOOL_PATH>`:
  - `<ATK_SRC_PATH>\gui_application\bin`
  - `<ATK_SRC_PATH>\gui_application\config`
  - `<ATK_SRC_PATH>\gui_application\image`

# Chapter 3

## Development Guidelines

This chapter introduces the APIs for each component.

### 3.1 APIs of Device Program

The device program integrates the RAM kernel, Flash library, and Fuse library. The device program is downloaded to external RAM through USB/UART according to the ROM bootstrap protocol.

#### 3.1.1 Device RAM Kernel

The device RAM kernel is the interface that communicates with the host. The device RAM kernel receives commands from the host and calls the corresponding library functions to execute the commands, and then sends the response to the host (where you can see it).

##### 3.1.1.1 RAM Kernel Protocol

This section describes the RAM kernel protocol. A device parses commands and sends responses by following the RAM kernel protocol.

Each stage in the RAM kernel protocol begins with a command issued by the host to the device, followed by a response from the device to the host. [Table 3-1](#) shows command/response package format, and [Table 3-2](#) describes the parameters.

**Table 3-1. Command/Response Package Format**

<b>COMMAND (16 bytes)</b>	0x06	0x06	cmd[15:0]	addr[31:0]	param1[31:0]	param2[31:16]	param2[15:8]	Param2[7:0]
<b>RESPONSE (8 bytes)</b>	ack[15:0]		csum[15:0]	len[31:0]				
<b>DATA</b>	Data[ ]							

**Table 3-2. RAM Kernel Protocol, Compact Format**

Term	Description
addr[31:0]	Flash offset to Flash address or fuse address
cmd[15:0]	Command ID
param1[31:0]	Flash erase/dump/program size

Table 3-2. RAM Kernel Protocol, Compact Format (continued)

Term	Description
Param2[7:0]	Different usages for different operations: For dumping, it is used to indicate following up operation For programming, it is used to indicate the file format For erasing, it is reserved.
Param2[15:8]	Different usages for different operations: For programming, it is used to indicate following up operation For dumping and erasing, it is reserved
param2[31:16]:	Reserved
ack[15:0]	Successful or error ID
len[31:0]	Data length
csum[15:0]	Checksum when dumping Flash data
data[]	Data, whose length is identified by len[31:0]

### 3.1.1.2 Common Commands/Responses

The information that follows describes the commands and responses supported by the current RAM kernel.

#### 3.1.1.2.1 Reset

<b>Command</b>	0x06	0x06	0x0201	<i>Reserved</i>	<i>Reserved</i>	<i>Reserved</i>	<i>Reserved</i>	<i>Reserved</i>
----------------	------	------	--------	-----------------	-----------------	-----------------	-----------------	-----------------

#### 3.1.1.2.2 Download

<b>Command</b>	0x06	0x06	0x0202	Download address	Size in bytes	<i>Reserved</i>	<i>Reserved</i>	<i>Reserved</i>
<b>Response</b>	OK/Error		<i>Reserved</i>	<i>Reserved</i>	—			

#### 3.1.1.2.3 Execute

<b>Command</b>	0x06	0x06	0x0203	Execute address	<i>Reserved</i>	<i>Reserved</i>	Reserved	Reserved
----------------	------	------	--------	-----------------	-----------------	-----------------	----------	----------

#### 3.1.1.2.4 Get version

<b>Command</b>	0x06	0x06	0x0204	<i>Reserved</i>	<i>Reserved</i>	<i>Reserved</i>	—	—
<b>Response</b>	0x00		i.MX processor ID	Model name length	—			
<b>Data</b>	Flash model name							

### 3.1.1.3 Flash Commands/Responses:

#### 3.1.1.3.1 Flash initialize

<b>Command</b>	0x06	0x06	0x0001	<i>Reserved</i>	<i>Reserved</i>	<i>Reserved</i>	Reserved	Reserved
<b>Response</b>	OK/Error		<i>Reserved</i>	<i>Reserved</i>	—			

#### 3.1.1.3.2 Flash erase

<b>Command</b>	0x06	0x06	0x0002	Offset address	Size in bytes	<i>Reserved</i>	Reserved	Reserved
<b>Response</b>	0x01		<i>Reserved</i>	Erase progress (finished size)	—			
<b>Response</b>	OK/Error		<i>Reserved</i>	<i>Reserved</i>	—			

#### 3.1.1.3.3 Flash dump

<b>Command</b>	0x06	0x06	0x0003	Offset address	Size in bytes	<i>Reserved</i>	Reserved	Follow up operation indication
<b>Response</b>	OK/Error		Checksum	Data length	—			
<b>Data</b>	Data							

#### 3.1.1.3.4 Flash program (page or block program)

<b>Command</b>	0x06	0x06	0x0004	Offset address	Size in bytes	<i>Reserved</i>	Follow up operation indication	File format
<b>Response</b>	OK/Error		Checksum	Data length	—			

#### 3.1.1.3.5 Flash program (un-boundary program)

<b>Command</b>	0x06	0x06	0x0005	Offset address	Size in bytes	<i>Reserved</i>	Follow up operation indication	File format
<b>Response</b>	OK/Error		Checksum	Data length	—			

### 3.1.1.3.6 Flash get capacity

<b>Command</b>	0x06	0x06	0x0006	<i>Reserved</i>	<i>Reserved</i>	<i>Reserved</i>	<i>Reserved</i>	<i>Reserved</i>
<b>Response</b>	OK/Error		<i>Reserved</i>	Data length	—			

### 3.1.1.4 Extend Command/Response:

#### 3.1.1.4.1 Switch from UART to USB connection

<b>Command</b>	0x06	0x06	0x0301	<i>Reserved</i>	<i>Reserved</i>	<i>Reserved</i>	Reserved	Reserved
<b>Response</b>	OK/Error		<i>Reserved</i>	<i>Reserved</i>	—			

#### 3.1.1.4.2 Swap BI SWAP Flag

<b>Command</b>	0x06	0x06	0x0302	<i>Reserved</i>	Size in bytes	<i>Reserved</i>	Reserved	Reserved
<b>Response</b>	OK/Error		<i>Reserved</i>	<i>Reserved</i>	—			

#### 3.1.1.4.3 Set BBT Flag

<b>Command</b>	0x06	0x06	0x0303	<i>Reserved</i>	Size in bytes	<i>Reserved</i>	Reserved	Reserved
<b>Response</b>	OK/Error		<i>Reserved</i>	<i>Reserved</i>	—			

#### 3.1.1.4.4 Set interleave Flag

<b>Command</b>	0x06	0x06	0x0304	<i>Reserved</i>	Size in bytes	<i>Reserved</i>	Reserved	Reserved
<b>Response</b>	OK/Error		<i>Reserved</i>	<i>Reserved</i>	—			

### 3.1.1.5 Source Code of Device RAM Kernel

The device RAM kernel source code is located under <ATK\_SRC\_PATH>\device\_program\ram\_kernel\src.

Table 3-3 describes the source files.

**Table 3-3. Source files for Device RAM Kernel**

File	Description
main.c	Contains the main functions of the device program: Parses commands from host to execute Flash/fuse/utility functions, then sends responses to the device.
Platform\mx**\src\channel.c	Channel function: supports USB/UART for different i.MX chips.
Platform\mx**\src\platform.c	Platform specific functions to retrieve chip ID, reset and so on.



## 3.1.2 Flash Library

The Flash library contains interfaces and source code.

### 3.1.2.1 Device Flash Interfaces

The Flash library provides general operations (dump/program/erase) for different storage devices (NOR/NAND/MMC). The device RAM kernel calls Flash interfaces to execute commands from the host side. [Table 3-4](#) identifies the interfaces of the Flash library and their corresponding commands.

**Table 3-4. APIs of the Device Flash Library**

Command	Flash API
Flash initial	<pre>s16 atk_Flash_lib_initial (void);</pre> Initializes the Flash library device, including check device vendor, id, and other physical parameters. Returns: If initialization is OK, returns RET_SUCCESS; otherwise returns error information.
Flash erase	<pre>s16 atk_Flash_lib_erase (u32 addr, u32 size,                           response_callback callback);</pre> Erases an area of Flash. Parameters: addr [in] Flash offset address where erasure will start size [in] erase size in bytes. callback [in] response send callback. Returns: If erase is successful, returns RET_SUCCESS; otherwise returns FLASH_FAILED;
Flash dump	<pre>s16 atk_Flash_lib_read (u32 addr,                         u8 *buffer,                         u32 count,                         dump_callback callback,                         u32 bufsize);</pre> Dumps an area of Flash and returns the data size of the area that was actually dumped. Parameters: addr [in] the Flash offset address where the area to be dumped starts buffer [out] an input buffer for library function to fill data in count [in] the dump size in bytes. callback [in] response send callback. bufsize [in] the size of buffer: If dumping is successful, returns RET_SUCCESS; if the dump range exceeds the Flash size, returns FLASH_PARTLY; if ECC error occurs, returns FLASH_ECC_FAILED; otherwise returns FLASH_FAILED.

**Table 3-4. APIs of the Device Flash Library (continued)**

Command	Flash API
Flash program	<pre>s16 atk_flash_lib_program (u32 addr,                            const u8 *buffer,                            u32 *pcount,                            u16 *pcsum,                            u8 mode,                            u8 file_format,                            response_callback callback);</pre> <p>Programs an area of Flash, and returns the actual programmed size and checksum.</p> <p>Parameters:</p> <p>addr [in] the Flash offset address where the area to be programmed starts.</p> <p>buffer [in] buffer containing the program data.</p> <p>pcount [in/out] program size in bytes, and returns the actually program size.</p> <p>pcsum [out] returns the actually programmed data checksum on Flash.</p> <p>mode [in] See FLASH_PRG_MODE: boundary or un-boundary mode.</p> <p>file_format[in] See FLASH_FILE_FORMAT: normal format(binary) or nb0 file format. It is only used by the NAND Flash driver.</p> <p>callback [in] response send callback.</p> <p>Returns:</p> <p>If programming is successful, returns RET_SUCCESS;</p> <p>if the selected program range exceeds the Flash size, returns FLASH_PARTLY;</p> <p>if ECC error, returns FLASH_ECC_FAILED;</p> <p>otherwise returns FLASH_FAILED.</p>
Get Model	<pre>void atk_flash_get_model(u8 *fmodel,                         u32 *len);</pre> <p>Gets the Flash name running on the target</p> <p>Parameters:</p> <p>fmodel [out] pointer to the buffer to store the Flash name.</p> <p>len [out] the Flash model length, which should be less than FLASH_MODEL_MAX.</p> <p>Returns:</p> <p>N/A</p> <p><b>Note:</b> The Flash model string returned here should be the same as the Flash model string defined in &lt;ATK_SRC_PATH&gt;\gui_application\config\ADSToolkit.cfg. Otherwise, the Flash tool may not work normally.</p>
Get Capacity	<pre>s16 atk_flash_get_capacity(u32 *size);</pre> <p>Get the flash capacity</p> <p>Parameters:</p> <p>size [out] pointer to the buffer to store the flash capacity</p> <p>Returns:</p> <p>If getting capacity is successful, returns RET_SUCCESS;</p> <p>otherwise return FLASH_FAILED</p>

**NOTE**

When supporting a new flash, you must re-implement the APIs in [Table 3-4](#). If a new flash is similar to the flash types supported by the ATK, you can use the source codes in [Section 3.1.2.2, “Source Code of Device Flash Library,”](#) as a reference.

### 3.1.2.2 Source Code of Device Flash Library

The source code of the device Flash library is located under <ATK\_SRC\_PATH>\device\_program\Flash.

Table 3-5 describes these source files.

**Table 3-5. Source Files of Device Flash Library**

File	Description
mmc_Flash\*	Files that support programming/dumping/erasing on MMC/SD card: <ul style="list-style-type: none"> <li>mx3x_mmc: supports i.MX31/i.MX32 MMC</li> <li>mx35_mmc: supports i.MX35 TO1/TO2 MMC/SD</li> <li>mx37_mmc: supports i.MX37 MMC/SD</li> <li>mx51_mmc: support i.MX51 MMC/SD</li> <li>mx25_mmc: support i.MX51 MMC/SD</li> </ul>
sd_Flash\*	Files that support programming/dumping/erasing on SD card: <ul style="list-style-type: none"> <li>mx3x_sd: support mx31/mx32 SD</li> </ul>
nor_Flash\*	Files that support programming/dumping/erasing on NOR Flash- spansion/inc: header files of nor flash and spansion nor flash low level driver file <ul style="list-style-type: none"> <li>spansion/src: source code of spansion nor flash operations and interfaces</li> </ul>
nand_Flash\*	Files that support programming/dumping/erasing on NAND Flash: <ul style="list-style-type: none"> <li>Inc: header files and low level driver operations for different NAND flashes</li> <li>src: source code of NAND flash operations and interfaces</li> </ul>
unit_test\*	Files that support a dummy Flash library

### 3.1.3 GNU Building Environment for Device Program

Source files that are specific for the GNU-building environment are located under <ATK\_SRC\_PATH>\device\_program.

Table 3-6 describes the files.

**Table 3-6. Source Files of GNU Environment for Device Program**

File	Description
Init\init.s	Start-up code for device program, which includes setting heap and stack, and jumping to main.
Makefile, rules.make, Flash\Makefile  Init\Makefile ram_kernel\Makefile	Makefiles for GNU-building environment, which define how to build a device program.
ram_kernel_mx31.lds ram_kernel_mx32.lds ram_kernel_mx27.lds ram_kernel_mx35.lds ram_kernel_mx37.lds ram_kernel_mx51.lds ram_kernel_mx25.lds	Scatter files for different chips. The start address of the device program is linked as 0x8000_4000 for i.MX31/i.MX32 boards, 0x0xA000_4000 for i.MX27 boards, 0x8000_4000 for i.MX35/i.MX37/i.MX25 boards, 0x9000_4000 for i.MX51 board. <b>Note:</b> The above 0x4000 bytes before link start address are used by the GUI application: When the memory layout in the device program is changed, please verify that the file MXDefine.h in <ATK_SRC_PATH>\gui_application\Platform\ is updated accordingly (if necessary). For i.MX35/i.MX37/i.MX51/i.MX25, the flash header should be reserved in the link file as ROM required. See i.MX35/i.MX37/i.MX51/i.MX25 IC Spec docs for more details.

## 3.2 APIs of ROM/RAM Kernel Host DLL

The ROM/RAM kernel Host DLL is a common component that communicates with the device program. It runs on the host and supports two protocols:

- ROM bootstrap protocol, which communicates with the ROM bootstrap.
- RAM kernel protocol, which communicates between host and device RAM.

The ROM/RAM kernel Host DLL provides a set of APIs that enables the GUI application to notify the device to execute corresponding operations.

### 3.2.1 Interfaces for ROM Kernel

The ROM bootstrap protocol enables you to download an application to one address and then execute that application in place. For more protocol information, see the *i.MX31 ROM User's Guide*. In the ATK environment, the ROM kernel protocol downloads the device program (RAM kernel, Flash library, and Fuse library) into external RAM. [Table 3-7](#) describes the DLL APIs for the ROM kernel.

**Table 3-7. APIs of Host ROM Kernel DLL**

API	Description
InitComPort	Initializes the serial port that the PC is using, Config = 0 (MX31 TO1); Config = 1, i.MX31 TO2/i.MX27 TO1, i.MX32, i.MX35, i.MX37, i.MX51, i.MX25
CloseComPort	Closes the Serial Port, Config = 0 (MX31 TO1); Config = 1, i.MX31 TO2/i.MX27 TO1, i.MX32, i.MX35, i.MX37, i.MX51, i.MX25
WriteMemory	Writes data to a given address in memory. This address must be mapped in the i.MX process memory space.
Jump2Rak	Issues a complete command and jump to the entry function of a device program, then starts to execute the device program in the device.
DownloadCSF	Downloads a CSF image to the given address.
DownloadDCD	Downloads a DCD image to the given address
DownloadImage	Downloads the binary image file to the given address
ReadDataByRok	Reads the data from the given address
GetHABStatus	Reads the HAB status return from the ROM when HAB couldn't pass authentication
OpenUSB	Opens a USB device
CloseUSB	Closes a USB device

### 3.2.2 Interfaces for Host RAM Kernel

The RAM Kernel Host DLL follows the protocol defined in [Section 3.1.1, "Device RAM Kernel,"](#) to assemble RAM Kernel commands and parse the responses. The RAM Kernel Host DLL provides an interface to the GUI application to execute flash and fuse operations.

Table 3-8 lists the APIs of the RAM Kernel Host DLL. Note that the parameters are similar to those in Section 3.1, “APIs of Device Program.”

**Table 3-8. APIs of Host RAM Kernel DLL**

API	Description
int InitFlash(void);	Initializes Flash.
int EraseFlash(unsigned long addr, unsigned long size);	Erases a select range of Flash. Supports erasing by block.
int ReadFlash(unsigned long addr, unsigned char *buffer, unsigned long count);	Dumps a selected area of Flash data.
ProgramFlash(unsigned long addr, const unsigned char *buffer, unsigned long count, int mode,UCHAR format =0)	Programs Flash.
int CommonReset(void);	Resets the device.
int GetRKLVersion(unsigned char *fmodel, int *len, int *mxType);	Gets device status (bootstrap or RAM Kernel): fmodel Flash model string len Flash model string length mxType chip ID Return: If the RAM kernel is running and len is not equal to 0, returns RET_SUCCESS.. If the bootstrap is running and len is equal to 0, returns RET_SUCCESS.. If no program is running, returns INVALID_CHANNEL.
int CommonDownload(unsigned long addr, unsigned long size, const unsigned char *pBuf)	Downloads data to one address using the RAM kernel protocol. addr the address to be downloaded to size data length pBuf data
int CommonExecute(unsigned long addr)	Jumps the device program to the specified address.
bool AtkHostApiClass::DoCom2Usb(void)	Request to switch from UART to USB
int AtkHostApiClass::SetBISwapFlag(int flag)	Set the BI Swap Flag
int AtkHostApiClass::SetBBTFlag(int flag)	Set the BBT Flag flag Input flag Return: return int,, none zero mean failed
int AtkHostApiClass::SetINTLVFlag(int flag)	Set the interleave Flag flag Input flag Return: return int,, none zero mean failed
int GetFlashCapacity(unsigned long *size)	Get flash capacity *size pointer to the flash capacity Return: Return int, none zero means failed

**NOTE**

The Host RAM Kernel APIs should be called when the RAM kernel is running (except for the `GetRKLVersion`), which implies that these APIs should be called after executing `Jump2Rak` successfully. Otherwise, the `INVALID_CHANNEL` error will be returned.

**3.2.3 Common APIs for Host DLL**

Table 3-9 describes the common APIs of the Host DLL.

**Table 3-9. Common APIs of the Host DLL**

APIs	Descriptions
<code>void SetWinHandle(HWND hWnd);</code>	Sets the Window handle in order to send a message to the UI.
<code>void SetUpChannel(int channel, int usbId);</code>	Sets up the channel mode.
<code>void SetEvtHandle(HANDLE hEvent);</code>	Sets the event handle.

**3.3 GUI Application—Use Cases**

This section introduces some common use cases in the GUI application. These use cases show how the GUI application implements Flash functions through the Host DLL and how the Host DLL communicates with the i.MX device.

### 3.3.1 UART Use Case

This use case describes how to download a device program through the UART.

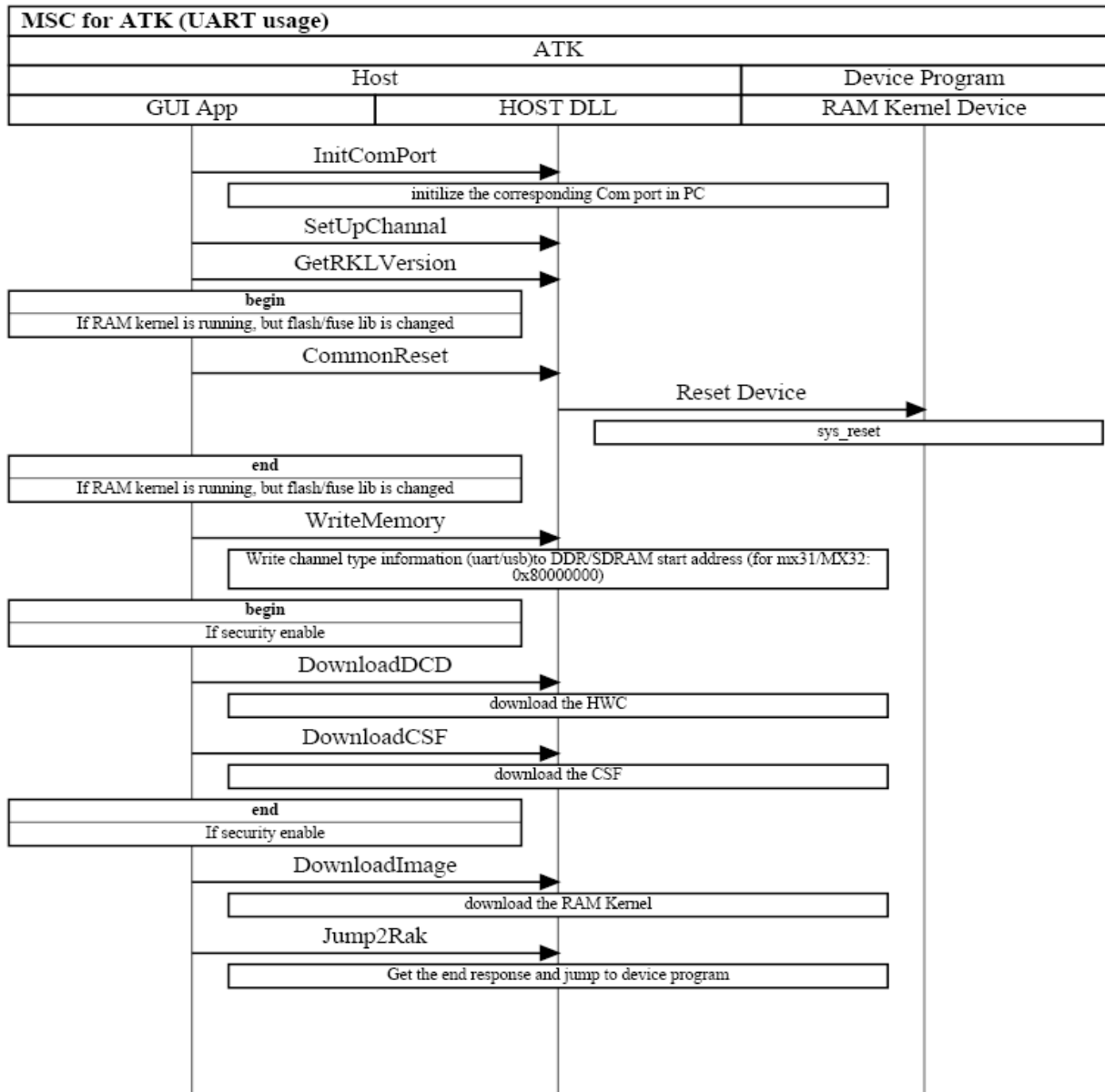


Figure 3-1. UART Use Case

### 3.3.2 USB Use Case

This use case describes how to download a device program through the USB.

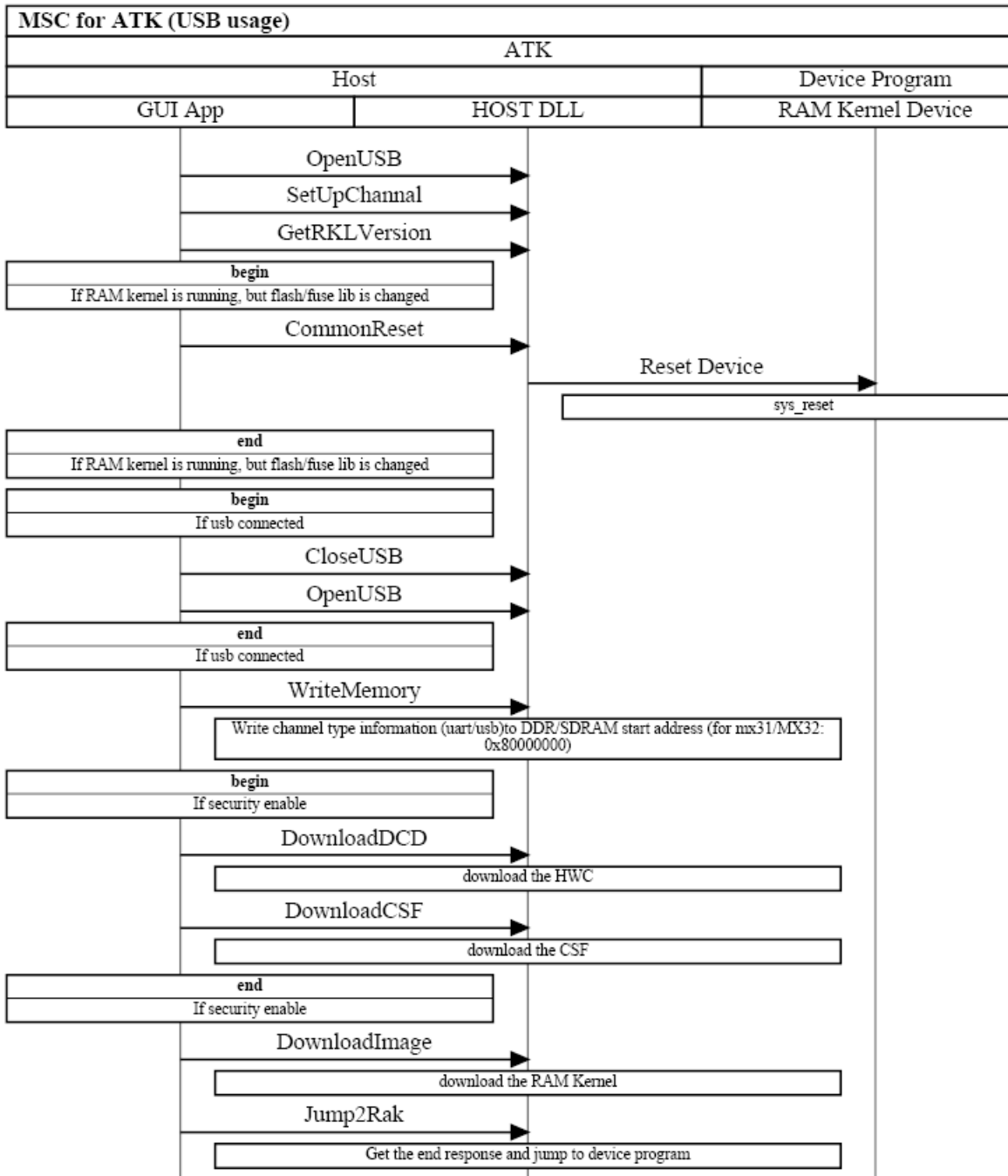


Figure 3-2. USB Use Case



### 3.3.3 Flash Programming Use Case

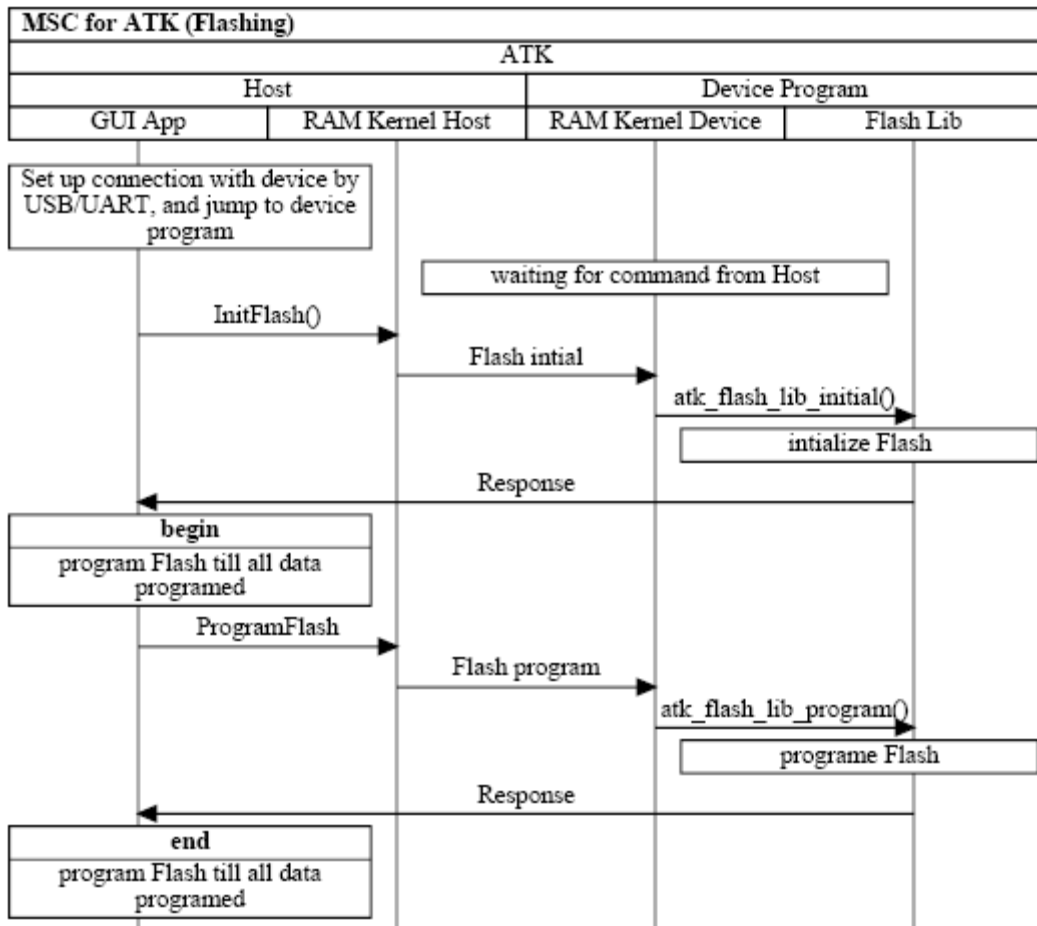


Figure 3-3. Flash Programming Use Case



## Chapter 4

### Additional Information

This chapter provides information about supporting a new Flash device, developing a device program using CodeWarrior, response for the Flash operations, and installing Cygwin for the ATK.

#### 4.1 Supporting a New Flash Device

##### 4.1.1 NAND Flash

Generate a new Flash library, using the following steps:

1. Add the new flash model to the `nand_ids.c` file. Correctly set each file.

```
/*man dev io ps oob mo po scan row blks ppb name */
{0xEC, 0x76, 8, 512, 16, 5, 0, 1, 3, 4096, 32, " myflashmodel " },
```

The `nand_ids.c` file is located at: `<ATK_SRC_PATH>/device_program/flash/nand_flash/src`

2. Build the new flash lib according to the target platform, for example:

```
make MCU=mx31 FLASH_TYPE=NAND
```

3. Copy the new flash lib to the location: `<ATK_SRC_PATH>/gui_application/image/`
4. Load a new Flash library into the GUI. Add the new flash model string to the file

`<ATK_SRC_PATH>/gui_application/config/ADSToolkit.cfg`

For example:

```
[Flash Model]
[MX31]
NAND:myflashmodel:image\mx31_nand.bin
```

## 4.1.2 NOR Flash

### 4.1.2.1 Generate new SPANSION NOR Flash Model

Generate a new Flash library, using the following steps:

1. Add the new flash model to the supported\_model array in `nor_flash.c` (`<ATK_SRC_PATH>/device_program/flash/nor_flash/spansion/src`) file, as follows:

```
static struct nor_flash_model supported_model[] = {
    {
        .device_name = "SG29GL512N",
        .device_size = DEVICE_64M,
        .max_wb_word = 16,
        .device_id = 0x22012223,
        .sector_size = { SECTOR_128K, 0 },
        .sector_mask = 0,
    },
};
```

**device name:** will be returned by the function `atk_Flash_get_model`

**device\_size:** in byte size

**max\_wb\_word:** max write buffer word

**device\_id:** NOR flash ID read out by autoselect mode

**sector\_size:** sector size array, list out the supported different sector sizes

**sector\_mask:** use sector address to mask this value, and get the sector size index in `sector_size[]`

2. Build the new flash lib according to the target platform. For example:

```
make MCU=mx31 FLASH_TYPE=NOR FLASH_MODEL=SPANSON
```

3. Copy the new flash lib to the location: `<ATK_SRC_PATH>/gui_application/image/`
4. Load a new Flash library into the GUI.
5. Add the new flash model string to the file

```
<ATK_SRC_PATH>/gui_application/config/ADSToolkit.cfg
```

For example:

```
[Flash Model]
[MX31]
NAND:myflashmodel:image\mx31_nor.bin
```

### 4.1.2.2 Support New Flash Model

1. Create your new flash model source folder under `<ATK_SRC_PATH>/device_program/flash/nor_flash/`
2. Create `inc/` `src/` directory, and put your source files and include files into them
3. To generate the new Flash library, type the command.

4. `make MCU=mx31 FLASH_TYPE=NOR FLASH_MODEL=YOURMODEL`
5. Make sure the flash library has been implemented and exported to the ram kernel. You do not need to modify the Makefile under `<ATK_SRC_PATH>/device_program/flash/`.

## 4.2 Developing a Device Program using CodeWarrior

There are examples for developing a device program using CodeWarrior in:

`<ATK_SRC_PATH>/device_program/cw_mcp`

To develop a device program using CodeWarrior, use these steps:

1. Create an mcp project file and add the necessary files to the project. [Table 4-1](#) describes the files specified by Code Warrior.

**Table 4-1. Source Files for the Device RAM Kernel**

File	Description
init.s	Startup codes for device program
heap.s stack.s retarget.c	Define stack and heap for device program
scat_ram	Scatter files

For other necessary files for Flash, fuse, and RAM kernels, see [Section 3.1.1, “Device RAM Kernel,”](#) through [Section 3.1.3, “GNU Building Environment for Device Program.”](#)

34. Select **Edit > Debug/Release Settings**. Set compiler/build parameters such as Target setting, access path, Language settings and Linker, and others.
35. Select **make** to build the device program.

## 4.3 Why the Flash Operations need Response Callback

The Flash operations may take a long time due to the performance of certain Flash chipsets, such as NOR FLASH erase. This is because the ATK host does not get a response during the **erase** progress, and therefore does not count the number of erased sectors or blocks.

Meanwhile, the host UART **read** function waits for the response, including the timeout that makes the ATK host not stable enough. Therefore, the response **send callback parameter** is added to the Flash erase interface.

On the i.MX35/i.MX37/i.MX51/i.MX25 platforms, the watchdog modules on the chip were enabled before the RAM Kernel loaded. Therefore, all the flash operations that take a long time to finish must reset the watchdog by calling response callback, in order to send the current operation status back to the host.

The Flash library implements must follow the existing ones to handle the response callback.

## 4.4 Installing Cygwin for the ATK

To install Cygwin, use these steps:

1. Download `setup.exe` from <http://www.cygwin.com/setup.exe>.
2. Run `setup.exe`.

The installation screen is displayed (Figure 4-1).

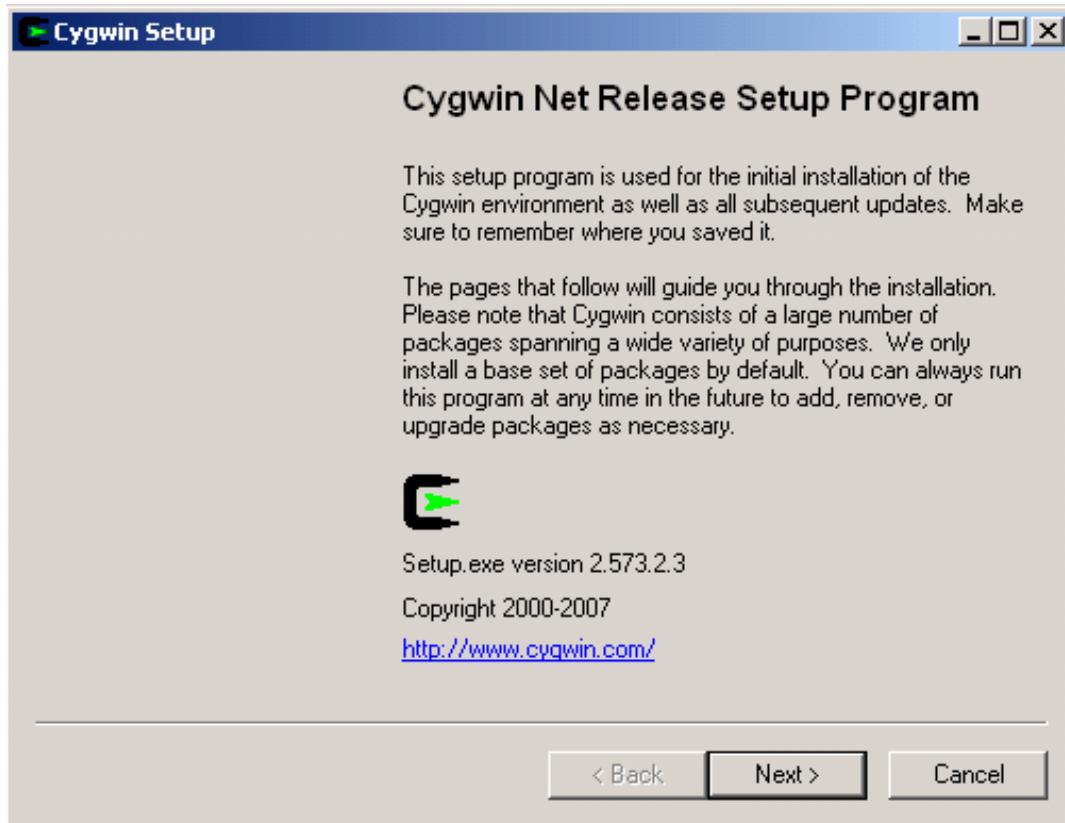
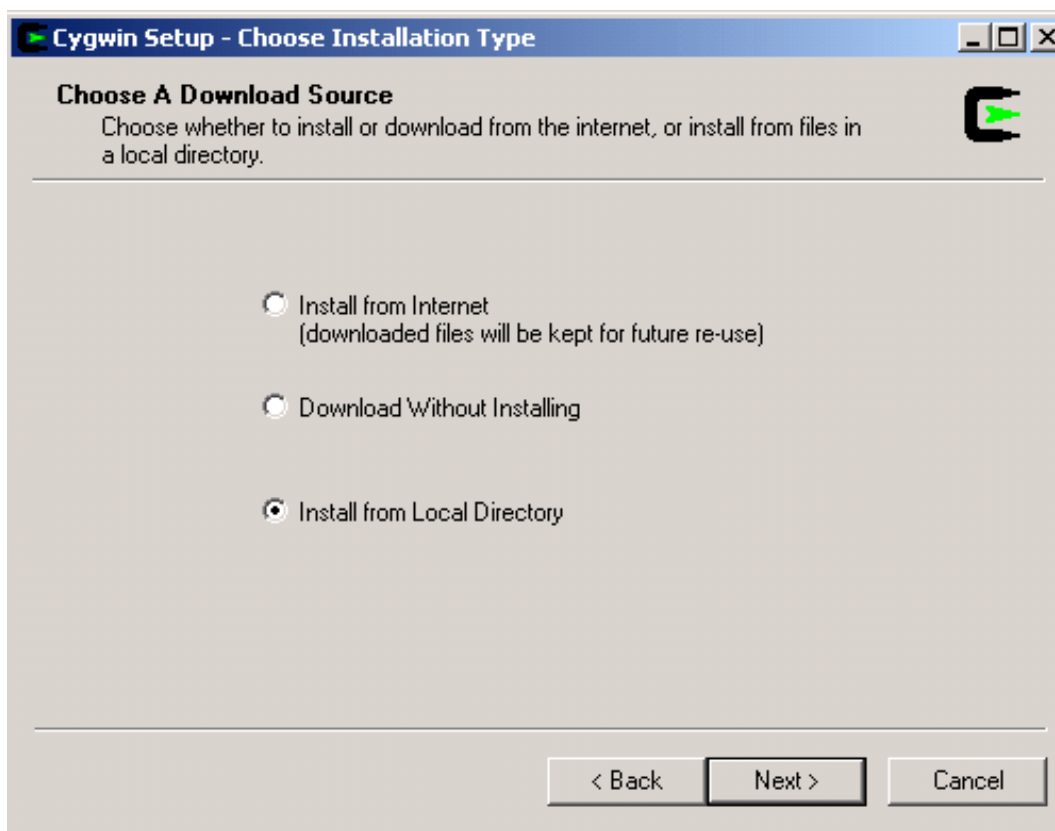


Figure 4-1. Installation Setup

3. Click **Next**, and prepare to select the installation method.

The download screen is displayed (Figure 4-2).



**Figure 4-2. Download Screen**

There are three installation methods:

- **Install from Internet:** Downloads from the Internet and installs immediately.
- **Download without Installing:** Downloads the installation package to your local directory but does not install it.
- **Install from Local Directory:** Does not download the installation package, but allows you to install the package from a local directory that already contains the package.

If you do not yet have the installation package, we recommend that you select **Download Without Installing**, and specify the installation path manually before clicking **Next**.

For the following example, the installation package is in the local directory, so we select **Install from Local Directory** and click **Next** to continue our example setup in step 4.

4. Specify the installation directory.

The Installation Directory screen is displayed (Figure 4-3).

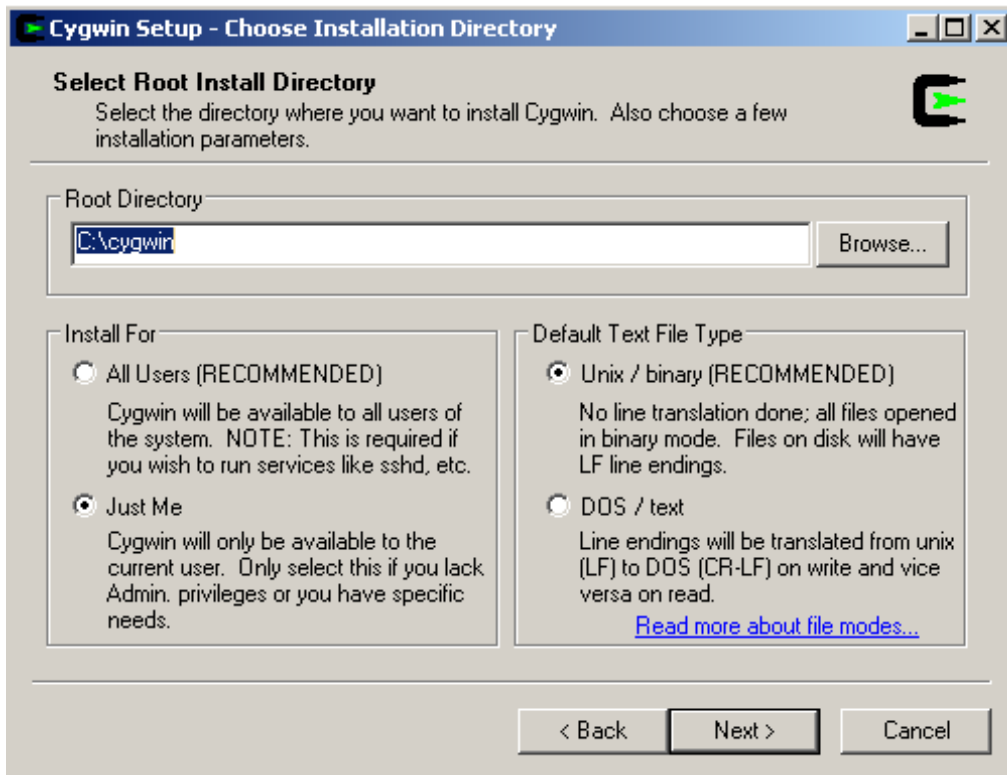


Figure 4-3. Installation Directory Screen

5. Select the local Package Directory.



The Local Package Directory screen is displayed (Figure 4-4).

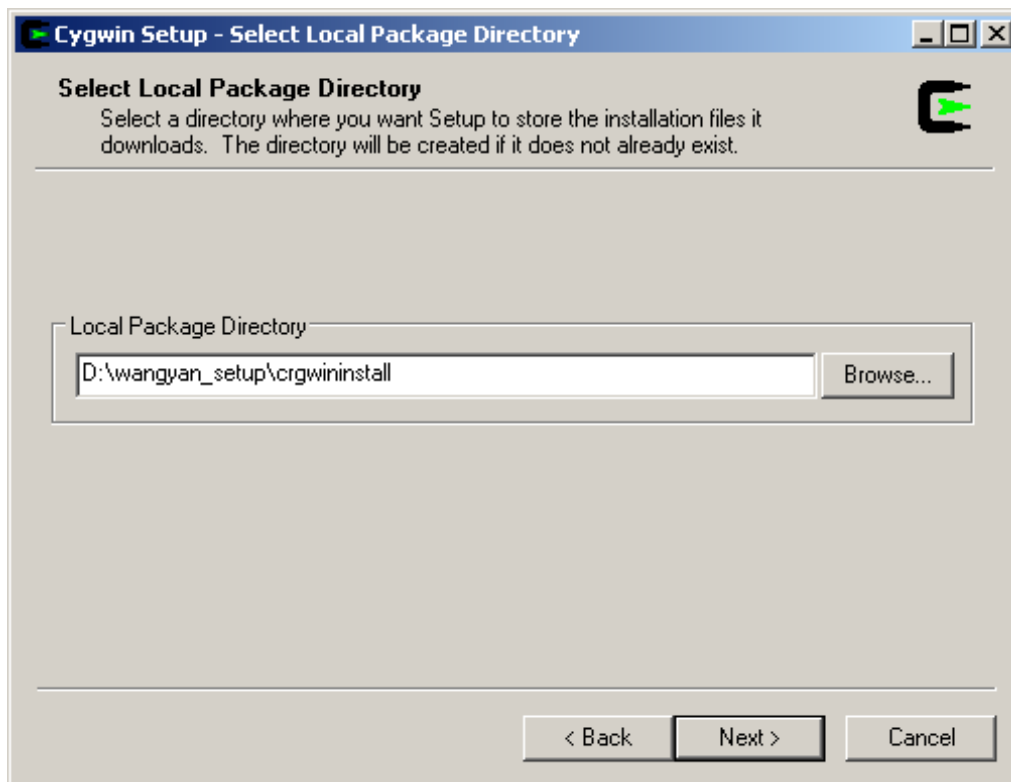


Figure 4-4. Local Directory Screen

Because we selected the installation method **Install from Local Directory**, we must specify the local directory.

6. Click **Next**.

The Select Package screen is displayed (Figure 4-5).

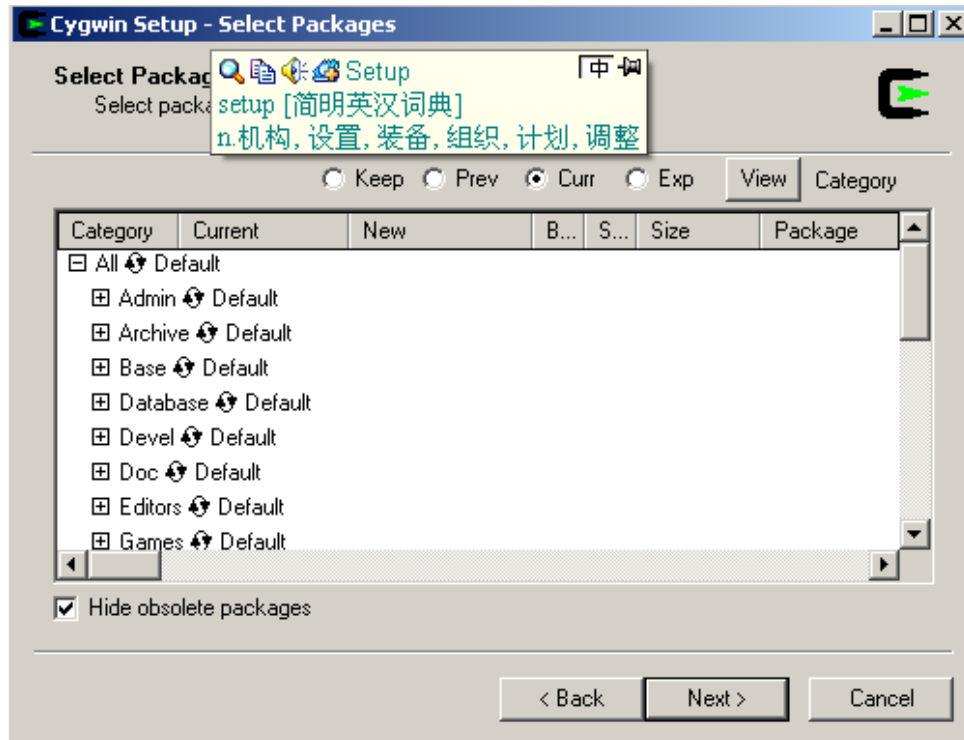


Figure 4-5. Select Package Screen

Options:

- **Default:** installs only the default installation items.
- **Install:** installs all, and requires about 200M space.
- **Reinstall:** Reinstalls the selected items.
- **Uninstall:** Uninstalls the selected items.

Select **Install** to install all.

For the ATK, the **make** option must be selected as “install”. “make” is under the “Devel” as shown in Figure 4-6.

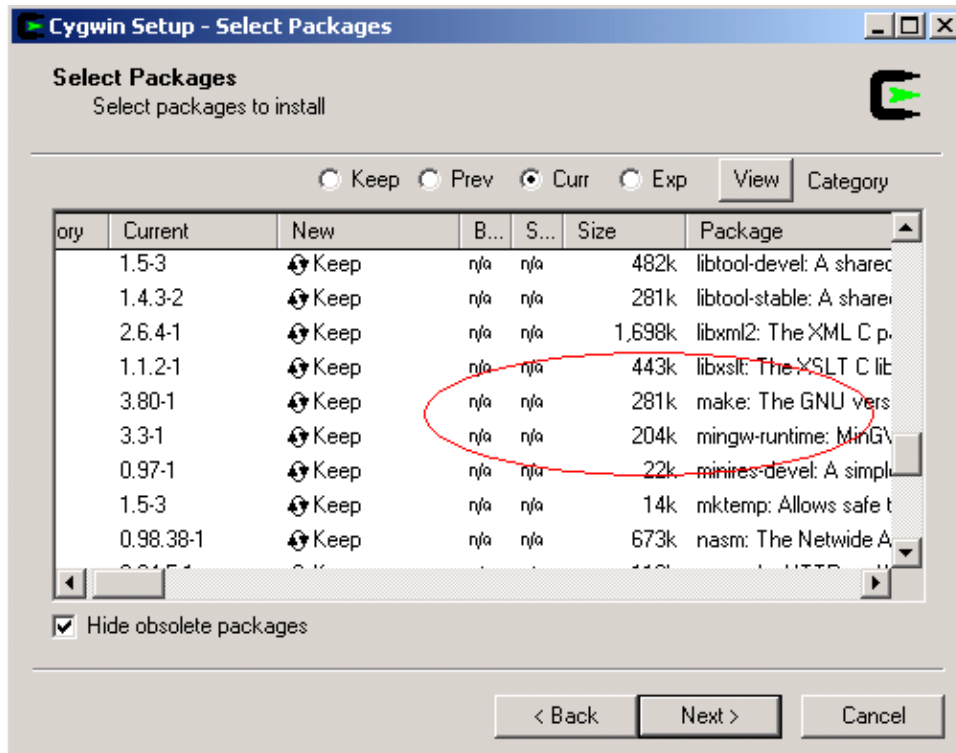


Figure 4-6. make

7. Click **Next**.

Installation begins (Figure 4-7).

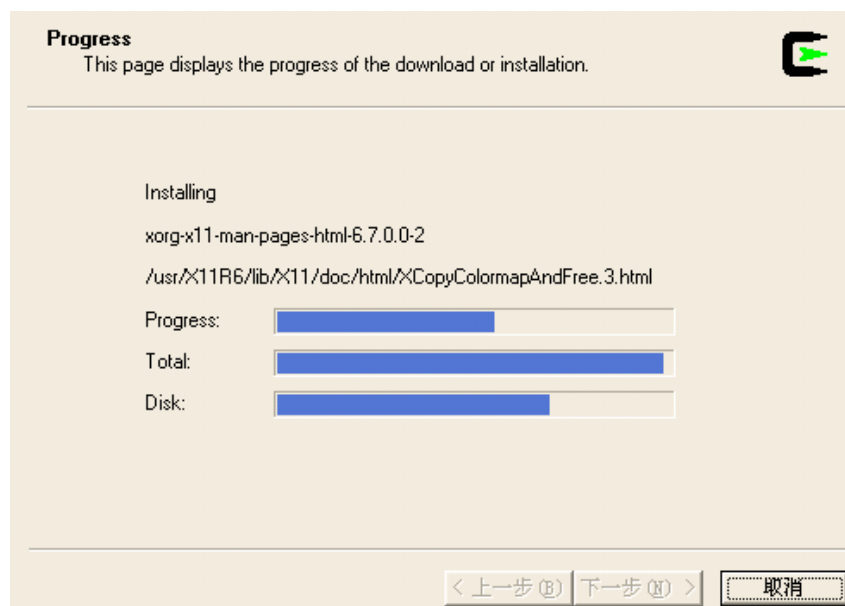


Figure 4-7. Installation

Installation takes about 30 minutes.

8. The next screen allows you to create icons (Figure 4-8).

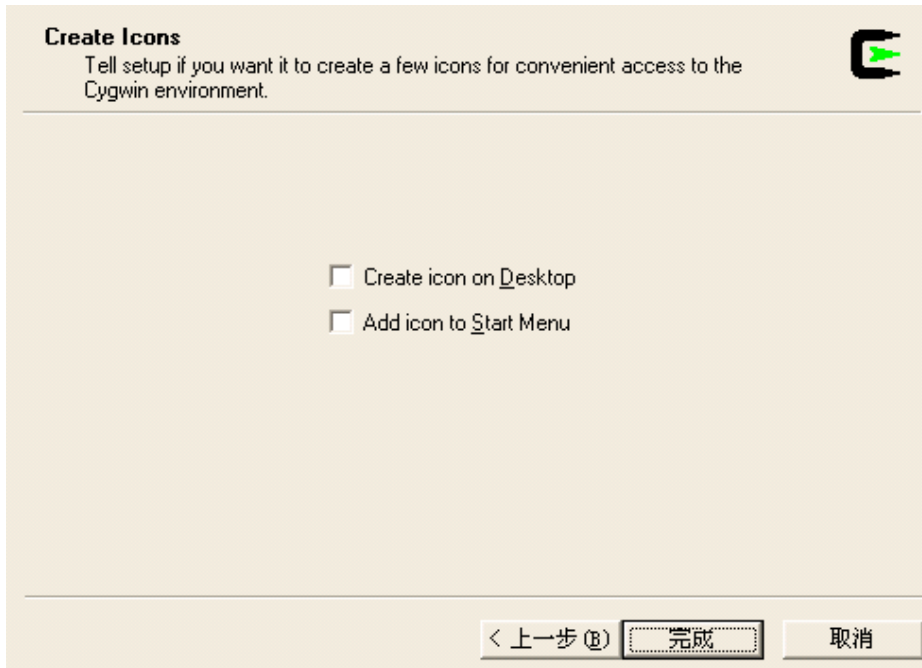


Figure 4-8. Creating Icons

The installation is complete (Figure 4-9).



Figure 4-9. Installed

Installation is successful. You may launch the Cygwin Bash shell and print “which make” to ensure that make is installed.