Design Brief 205
Understanding the Anadigm® Boot Kernel (ABK)

Introduction
This document describes the Anadigm® Boot Kernel (ABK) and the ABK command language.

Overview
The Anadigm® Boot Kernel is a set of software requirements and communication protocols designed to allow a PC running AnadigmDesigner®2, or a user application, to program and control an Anadigm® FPAA. The ABK provides an interface between PC and FPAA.

The ABK is provided as a core set of source code files in the C programming language. The source files can be adapted as needed to meet the requirements of the design. Additional features can be added as desired, and ABK functionality may be removed, depending on the capabilities of the target environment. The ABK source code may also be used as a basis for creating embedded software BUT it is important to note that the ABK itself is not required for an embedded system.

The ABK is not meant to be the core software of a target controlling the Anadigm® FPAA. Rather, the ABK provides a quick means of prototyping hardware designs before committing to the real application.

IMPORTANT: the ABK is not required for an embedded system.

ABK Environment
When a target runs the ABK, it creates a working environment for communications between the AnadigmDesigner®2 software (or other PC application) and the Anadigm® FPAA. This is illustrated in Figure 1.

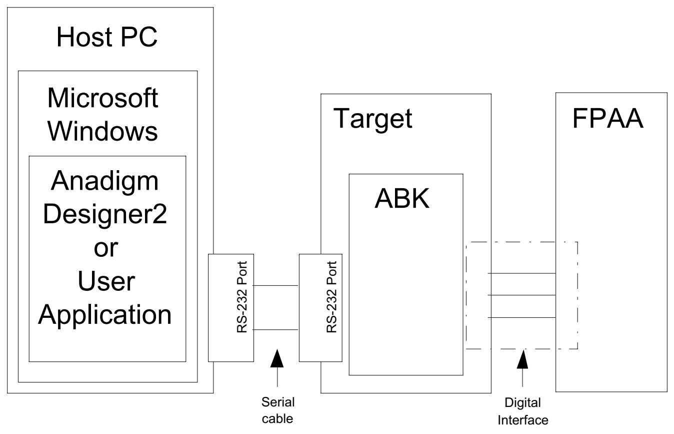

Figure 1. Basic ABK Environment

ABK Capabilities
The ABK defines a set of capabilities required to allow communication between the PC and the FPAA(s). These features are:

1. Primary configuration of one or more FPAAs
2. Reconfiguration of one or more FPAAs
3. Resetting of individual or groups of FPAAs
4. Reading the target's ADC channels
5. Reading the target's SPI port
6. Returning target ID and version information
7. Reporting operational & programming status
8. Displaying FPAA programming status via LEDs
9. Identification of FPAA types and number
10. Returning ABK's available command set

These features can be used in whole or in part, depending on the capabilities of the target. For example, feature 4 would have to be omitted if the target did not have analog-to-digital conversion circuitry, and feature 5 would have to be omitted if the target did not have a SPI port.

At a basic level, a target must support features 1, 6 and 7 in order for AnadigmDesigner to communicate with it and download configurations.

ABK Command Language
The ABK Command Language provides a uniform means of using the ABK features from AnadigmDesigner®2 or other applications. It is an ASCII-based command language, which allows easier command synchronization, error checking, and simplified command structure.

The commands in the ABK Command Language, their parameters and structure, are described in the Appendix of this document.

ABK Feature – Configuration Download
When a design is created within AnadigmDesigner®2, this feature allows the design to be downloaded into one or more FPAA devices.

There are 2 forms of configuration download:- the primary configuration which is preceded by an FPAA reset, and the reconfiguration which is not preceded by an FPAA reset and which can be used to perform partial updates of the FPAA

ABK Feature – Analog-to-Digital Readings
For targets that have Analog-to-Digital converters (ADC), this feature is useful for reading an A/D channel and reporting the result.

AnadigmDesigner®2 does not use this feature, and thus is not required for ABK implementation.

ABK Feature – SPI Port Readings
For targets that have a SPI port, this feature is useful for reading the SPI port and reporting the result. The SPI port can be used to read the SAR ADC in an FPAA.

AnadigmDesigner®2 does not use this feature, and thus is not required for ABK implementation.

ABK Feature – Target Identification and Version Information
This feature is useful for a variety of reasons, but the main one is communications tests. AnadigmDesigner®2 has a menu item, "Target->Display board information", which asks the target to identify itself.

AnadigmDesigner®2 will display the null-terminated string reported by the target, but does not make any further use of it.

The exact contents of the string are left to the designer, but some suggestions are:

- Board identifier
- Software and/or Hardware version number
- Revision Date of Software/Hardware

ABK Feature – Chip Identification
This feature is used to identify how many FPAA devices are connected to the target, and what types of FPAA they are.

This feature is used by AnadigmDesigner®2 which has a menu item, "Target->Display board information", which asks the target to identify itself and also to return the types of chips in the system.

ABK Feature – Operational and FPAA Programming Status
When configuration data is downloaded, AnadigmDesigner®2 will request status information from the target, in order to determine whether the download succeeded or not. If the status indicates a problem, AnadigmDesigner®2 will report that result to the user.

The target returns one ASCII character, which describes the current status of the target.

Note: the ABK also drives a red LED and a green LED to provide the user with instant feedback of programming status.

ABK Feature – Resetting of device(s)
The ABK can reset individual or multiple FPAAs through this command. When the command is used, the parameter specifies the ID of the FPAA(s) to be reset.

A specific FPAA can be reset by using its primary ID, assigned in AnadigmDesigner®2. A group of devices can be reset by assigning them a common secondary ID, and using that ID for the reset parameter. All devices can be reset when 0xFF is used for the ID.

ABK Feature – Command Set
This feature is used to allow AnadigmDesigner®2 to know what types of commands the target ABK will support. The returning string includes all commands supported.

ABK Feature – Pin Status Check
This feature is used to check the state of either the ACTIVATE pin or the ERRb pin. The Pin Status command 'P' needs to be followed by a Status check '?' to complete the checking of the state of the pin.

Appendix – The ABK Command Language

This appendix describes the ABK Command Language in detail.

ABK Command Language structure

The primary structure of the command language is shown in Figure 2:

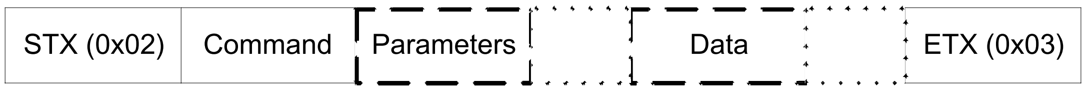

Figure 2 – ABK Command Language structure

Each command starts with the ASCII STX character, defined as the hex value 0x02, and ends with the ASCII ETX character, defined as the hex value 0x03. The Command character depends on the action desired of the ABK. Parameters and Data will depend on the command and whether the command requires parameters or data or not. All parameters and data are given in ASCII format. For each byte of data, two ASCII characters are required. For example, if a Data byte value was hex 0xA5, it would be sent as two ASCII characters, ‘A’ and ‘5’. For word values (e.g. 0xE534), four ASCII characters are required (e.g. ‘E’, ‘5’, ‘3’, ‘4’). All values are assumed to be unsigned, and any hex values must use upper-case letters (‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’) instead of lower-case letters.

Table 1 lists each command with the Parameters and Data (if any) required by that command.

Table 1 - ABK Commands

Feature	Command	Parameters	Data	Notes
Download Configuration Data	‘0’	One word for the number of Data bytes to follow.	The configuration information to be forwarded to the FPAA(s)	This command will issue a full reset to all devices before sending the Data bytes to the FPAA(s).
	‘1’	One word for the number of Data bytes to follow.	The configuration information to be forwarded to the FPAA(s)	This command will **NOT** issue a full reset to the devices. This is useful for sending partial update information to the FPAA(s).
Device Reset	‘R’	One byte for the FPAA ID to be reset, or 0xFF for all FPAAs	None	If the same primary or secondary IDs are assigned to a group of FPAAs, then all can be reset using the one ID.
Pin Status	‘P’	Two bytes for the pin and its state:- Byte 1: ‘A’ or ‘E’ for ACTIVATE or ERRb Byte 2: ‘H’ or ‘L’ for High or Low	None (see Notes)	This command should be followed by a ‘?’ Status check. If the Status check returns ‘0’ (OK) then the pin status is correct, if it returns ‘1’ (error) then the pin status is wrong. e.g. ‘P’ followed by ‘A’ and H’ followed by status check which returns ‘1’ means that the ACTIVATE pin is low.
Target ID and Version info	‘#’	None	None	The target returns a null-terminated string with the desired information. AnadigmDesigner2® will display this information in a dialog box when the “Target->Display board information” menu item is selected.
Status	‘?’	None	None	Target returns an ASCII character representing the status, with the following values: ’0’ – OK ’1’ – Configuration error ‘4’ – Unknown command
Analog-to-Digital Reading	‘V’	ASCII character for ADC channel to read. i.e. ‘0’ for channel 0, ‘1’ for channel 1, etc.	None	Target returns value of ADC for desired channel. Data size may depend on ADC capabilities.
SPI-port Reading	‘W’	None	None	Target returns value of SPI port. This is only available on ABK v3.1 and later Note: this may be used for reading the SAR ADC of the FPAA
Chip identification	‘T’	None	None	Target returns a null-terminated string that gives the types of FPAA(s) in the system. Possible identifiers are: ’220E04’ – AN220E04 ’221E04’ – AN221E04 ’120E04’ – AN120E04 ’121E04’ – AN121E04 Each chip type is separated by a vertical bar ‘|’ character. When all chips have been listed, the ABK sends an ‘END’. If a chip cannot be identified, it is reported as ‘UNKNOWN’, and identification stops.
Command Set	‘S’	None	None	Target returns a null-terminated string that lists the commands that are supported by the ABK. For example, a target that supports all ABK commands in this table would return a string like this: ‘01PRSTVW?#’

For more information logon to:
www.anadigm.com