

Report No:

AN111

Title:

ConsoleEDS Protocol for Remote Control of Equinox Programmers

Author:

John Marriott

Date:

10th July 2010

Version Number:

1.32

Console Application...



...for controlling Equinox ISP Programmers.

All rights are 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 prior 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

Contents

1.0 General overview of ConsoleEDS	8
1.1 Introduction.....	8
1.2 Features	9
1.3 Applications of the ConsoleEDS protocol.....	10
1.4 Advantages of a using a 'Console Application'	10
1.5 Limitations of the ConsoleEDS protocol.....	10
1.6 Programmers supporting ConsoleEDS	11
1.7 PC control of programmer via RS-232	11
1.8 Device (Target IC) programming support.....	12
1.9 Typical ConsoleEDS control example	13
1.10 Using ConsoleEDS to program unique Serial Numbers.....	14
1.11 Using ConsoleEDS to program unique Calibration Data.....	14
1.12 Using ConsoleEDS from the Command Window	15
1.13 Using ConsoleEDS from a batch file	15
2.0 ConsoleEDS product versions	16
2.1 Overview	16
2.2 Development Pack and Run-time license	16
2.3 Comparison of ConsoleEDS versions	17
2.4 Software Utilities included in each version of ConsoleEDS.....	18
2.5 ConsoleEDS - EVAL	18
2.6 ConsoleEDS STANDARD	19
2.7 ConsoleEDS - PRO	19
2.8 ConsoleEDS Multi-Port	20
2.9 ConsoleEDS Multi-Node	20
2.10 Which version of ConsoleEDS is my programmer enabled for?	21
2.0 ConsoleEDS Installation Instructions.....	22
2.1 Overview	22
2.2 Installation instructions	22
2.3 Installation Directory Structure	23
2.4 Running ConsoleEDS as a Standalone Application.....	24
2.5 Moving ConsoleEDS application to your "Working Directory"	24
3.0 ConsoleEDS – Getting Started Guide	26
3.1 Overview	26
3.2 Testing ConsoleEDS installation	26
3.3 Setting the Communications Parameters.....	27
3.4 Reading the Programmer Information	28
3.5 ConsoleEDS Command Format.....	29
3.6 Reading the 'Device ID (Signature)' of a Target IC	29
4.0 ConsoleEDS Command Specification	30
4.1 Overview	30
4.2 ConsoleEDS Command Format examples.....	30
4.3 Programmer Setup Commands.....	31
4.3.1 Overview	31
4.3.2 /SETUP command	32
4.3.3 /COMPORT command.....	33
4.3.4 /BAUDRATE command.....	33
4.3.5 /PROGINFO command	34
4.3.6 /CHECKMINIMUMBUILD command.....	34
4.3.7 /CHECKMINIMUMFIRMWARE command.....	35
4.4 Using ConsoleEDS with multiple programmers.....	36
4.4.1 Overview of using multiple programmers.....	36
4.4.2 Setting the unique 'Node Address' on a PPM3-MK2 or PPM4-MK1	37

4.4.3 How ConsoleEDS deals with multiple programmers	37
4.4.4 /DETECT command	39
4.4.5 /NODE command	40
4.4.6 /NODES command	41
4.4.7 /AUTOPROGRAMBROADCAST command	42
4.4.8 /TIMEBEFOREPOLL command	43
4.4.9 /TOTALTIMEOUT command	43
4.5 Standalone Programming Projects commands	44
4.5.1 Overview	44
4.5.2 /UPLOAD command	45
4.5.3 /PROJECTLIST command	46
4.5.4 /READPROJECTSTO command	46
4.5.5 /GETPROJECTSFROM command	47
4.5.6 /CHECKPROJECT command	48
4.5.7 /AUTOPROGRAM command	49
4.5.8 /GOIMMEDIATEPROG command	50
4.6 Power control + Enter / Exit Programming Mode commands	51
4.6.1 Overview	51
4.6.2 /POWERUP command	51
4.6.3 /RESET command	52
4.6.4 /NORESET command	52
4.6.5 /FASTSTART command	52
4.6.6 /AUTO_INIT command	53
4.7 Checking the Device Signature and JTAG ID	54
4.7.1 Overview	54
4.7.2 /READSIG command (Device Signature)	54
4.7.3 /READSIG command (Device JTAG ID)	55
4.8 /ERASE command	56
4.9 FLASH Area – WRITE, READ and VERIFY commands	57
4.9.1 Overview	57
4.9.2 /FLASHWRITE (entire file) command	57
4.9.3 /FLASHWRITE (specified address range) command	58
4.9.4 /FLASHVERIFY (entire file) command	58
4.9.5 /FLASHVERIFY (specified address range) command	58
4.9.6 /FLASHREAD (specified address range) command	59
4.9.7 /RPLB (AT91SAM7 - Read FLASH Page Lock Bits) command	59
4.10 EEPROM Area – WRITE, READ and VERIFY commands	60
4.10.1. Overview	60
4.10.2 /EEPROMWRITE (entire file) command	60
4.10.3 /EEPROMWRITE (specified address range) command	61
4.10.4 /EEPROMVERIFY (entire file) command	61
4.10.5 /EEPROMVERIFY (specified address range) command	62
4.10.6 /EEPROMREAD (specified address range) command	62
4.11 Programming 'Configuration Fuses' commands	63
4.11.1 Overview	63
4.11.2 /WRITE....FUSES command using a 'Fuse File'	63
4.11.3 /WRITE....FUSES command using 'HEX Fuse Values'	64
4.11.4 /FUSEWRITE command using 'HEX Fuse Values' (build 2008)	64
4.11.5 /READFUSES command	65
4.12 Security Fuse (Lock Bits) programming commands	66
4.12.1 Overview	66
4.12.2 /WRITESECURITY command using a Fuse File	66
4.12.3 /WRITESECURITY command using 'Hex Fuse Bytes'	66
4.12.4 /READSECURITY command	68
4.13 /READCAL command (AVR microcontrollers only)	69

4.14 Zensys ZWxxx – Read / Write HomeID command.....	70
4.14.1 Overview	70
4.14.2 /SETHOMEID command.....	70
4.14.3 /READHOMEID command.....	70
4.15 Scripting commands.....	71
4.15.1 Overview	71
4.15.2 /SCRIPT command	71
4.15.3 /INSERT command.....	71
4.15.4 /NOWAITKEY command.....	72
4.16 /SETINC command	72
4.17 /TIME command.....	73
4.18 /MEASUREVOLTAGE command.....	74
4.19 /MEASURECURRENT command	74
5.0 Creating Projects for ConsoleEDS	76
5.1 Overview	76
5.2 Creating a Base Project using EQTools – EDS	76
5.2.1 Creating an EDS – Development Project.....	76
5.2.2 Testing your EDS – Development Project with the Target IC.....	77
5.2.3 Exporting your EDS Project to Project Builder.....	78
6.0 Standalone Programming Projects	80
6.1 Overview	80
6.2 Advantages of Standalone Projects	80
6.3 Creating a Standalone Project for ConsoleEDS.....	80
6.3.1 Defining a ConsoleEDS project	80
6.3.2 Choosing a unique Project Name	81
6.3.3 Selecting a Standalone Project.....	81
6.4 Uploading Standalone Projects to a Programmer	82
6.4.1 Overview	82
6.4.2 Uploading a Project Collection to a Programmer.....	83
6.4.3 Uploading a Single Project to a Programmer.....	84
6.5 Downloading a Project List from a Programmer.....	85
6.6 Downloading a Project List from a Programmer to a file	85
6.7 Checking a Project is resident in the programmer.....	86
7.0 Typical ConsoleEDS control sessions	88
7.1 Typical ConsoleEDS Initialisation.....	88
7.2 Uploading a Project Collection	88
7.3 Executing a Standalone Project	89
7.4 Writing a BLOCK of data to a specified address range.....	89
7.5 Reading the entire EEPROM area back to a file	90
7.6 Reading a BLOCK of data from a specified address range back to a file	91
7.7 Programming the Configuration Fuses of the Target IC.....	91
7.7.1 Programming Fuses from a Project	91
7.7.2 Programming Fuses using the Hex Fuse Values.....	92
7.8 Programming the Security Fuses of the Target IC	92
7.8.1 Programming Fuses from a project.....	92
7.8.2 Programming Fuses using the Hex value(s).....	92
Appendix 1 – ConsoleEDS Error Code and Error reporting	94
1.0 Overview	94
1.1 ConsoleEDS – Exit Code / Error Code.....	94
1.2 ConsoleEDS – Error Code	94
1.3 ConsoleEDS – Batch File - Error Code example.....	95
Appendix 2 – ConsoleEDS Error Logging	96
1.0 Overview	96
1.1 Using the /LOG command	96
1.2 Using the /SETSTARTUPLOG command	96

Appendix 3 - Programmer Error numbers.....	97
--	----

1.0 General overview of ConsoleEDS

1.1 Introduction

This application note describes the '**ConsoleEDS**' protocol for remote control of Equinox programmers. **ConsoleEDS** is a powerful command-line orientated '**Console Application**' which can be used to control both low-level and high-level programming functions of any supported Equinox programmer. Sequences of commands can be executed from a '**Command Window**' allowing the programmer to be controlled manually, via DOS Batch File or via a custom '**Remote Application**'.

ConsoleEDS is ideal for controlling Equinox programmers from any '**Remote Application**' which runs under Windows and which is capable of shelling out to a '**Console Application**'. The command scripts can be sent directly from the '**Remote Application**' or they can be executed from a simple text file. This makes it possible to develop and test the control functionality independently of the '**Remote Application**' which can make the debugging process of the overall system much simpler.

ConsoleEDS supports higher level functions such as uploading and executing '**Standalone Programming Projects**'. It also supports the measurement of the target Voltage / Current, programming of FLASH / EEPROM, programming of Configuration Fuses and programming of Security fuses. As the **ConsoleEDS** commands are identical for each programmer / target IC, it is very straightforward to move **ConsoleEDS** control scripts from one programmer to another or from one target IC to another.

ConsoleEDS is a '**Console Application**' running under Microsoft Windows. This means that it can be quickly evaluated by launching a '**Command Window**' within any Windows session. It also makes it very simple to send commands from any other application written in e.g. C++, Delphi, .NET, Visual Basic, Labview etc.

The '**multi-node**' version of **ConsoleEDS** supports controlling of up to 32 x PPM4-MK1 or ISPnano programmers from a single session. It is possible to use a single command to automate the programming of entire panels of PCBs.

1.2 Features

ConsoleEDS is a powerful software utility which allows Equinox ISP Programmers to be controlled from a custom '**Remote Application**' running under Microsoft Windows.

The main features of ConsoleEDS are:

- Allows any Equinox ISP Programmer to be remote controlled via simple **Command Line** commands.
- ConsoleEDS control commands are independent of programmer type making it straightforward to port from one programmer to another.
- Suitable for interfacing to any application which executes under Microsoft Windows
- Allows the programmer to be controlled from a simple '**batch file**'
- Simple **Command Line** interface makes even complex programming operations simple to implement
- Supports controlling of both single programmers and also up to **32 networked programmers**
- Supports writing of a block of data from a file to the FLASH or EEPROM of a Target IC
- Supports reading of a block of data from the FLASH or EEPROM of the Target IC to a file on the PC hard disk
- Supports uploading of pre-compiled standalone '**Standalone Programming Projects**' to a target programmer without requiring EQTools or the Project Upload Wizard utility.
- Supports execution of a specified '**Programming Project**' which has already been uploaded into the programmer '**FLASH Memory Store**'
- Supports programming of the '**Configuration Fuses**' of the Target IC
- Supports programming of the '**Security Fuses**' of the Target IC
- Supports automatic generation of and programming of unique information such as **serial numbers** and **calibration data**
- Supports reading / writing of Atmel AVR on-chip '**Oscillator Calibration Byte**'
- Supports controlling of Target Power (e.g. during functional test)

Please note:

Many of the above features are only accessible when using either the '**Standard**' or '**Professional**' versions of 'ConsoleEDS. These are chargeable upgrades – for details please see Section 2.

1.3 Applications of the ConsoleEDS protocol

ConsoleEDS is a powerful software utility which allows any Equinox ISP Programmer to be controlled from a '**batch file**' or custom '**Remote Application**' running under Microsoft Windows.

The main applications of **ConsoleEDS** are:

- Programming any programmable IC
- Interfacing a programmer to any external third party control application
- Programming an **Intel HEX** or **Binary** data file into the FLASH or EEPROM of a target IC
- Programming unique '**Calibration Data**' into the FLASH or EEPROM of a target IC
- Programming a unique '**Serial Number**' or '**MAC Address**' into the FLASH or EEPROM of a target IC

1.4 Advantages of a using a 'Console Application'

The advantages of using a '**Console Application**' like **ConsoleEDS** instead of a '**Windows dll**' to control a programmer are as follows:

- Works on all versions of Windows without any changes
- Simple to test during the development phase as commands can be manually typed from the '**Command Window**'.
- Simple to debug if an implementation problem is found between **ConsoleEDS** and the '**Remote Application**' which is controlling **ConsoleEDS**.
- **ConsoleEDS** command sequences can be stored in '**batch files**' and recalled from any other application.

1.5 Limitations of the ConsoleEDS protocol

ConsoleEDS has been designed as a simple method to allow any '**Remote Application**' running under Windows to control the basic operations of any Equinox programmer.

The limitations of this protocol are as follows:

1. **ConsoleEDS** only works under **Microsoft Windows**. It will not work under '**MS DOS**'.
2. **ConsoleEDS** requires the use of a PC. It can therefore not be used on an '**In-Circuit Tester**' (ICT) unless the ICT is running a version of **Microsoft Windows**.
3. Although multiple programmers can be started at the same time using **ConsoleEDS**, it is not possible to sequence multiple programmers connected on the RS485 network to do different actions at the same time. This functionality is only available in our **ISP-PRO** control application.
4. **ConsoleEDS** is not a '**standalone**' application. It needs to be controlled from another application so some knowledge of a programming language is required. One way around this is to store ConsoleEDS command lists in a so-called '**batch file**'.

1.6 Programmers supporting ConsoleEDS

The Equinox programmers which are capable of supporting **ConsoleEDS** control are listed in the table below.

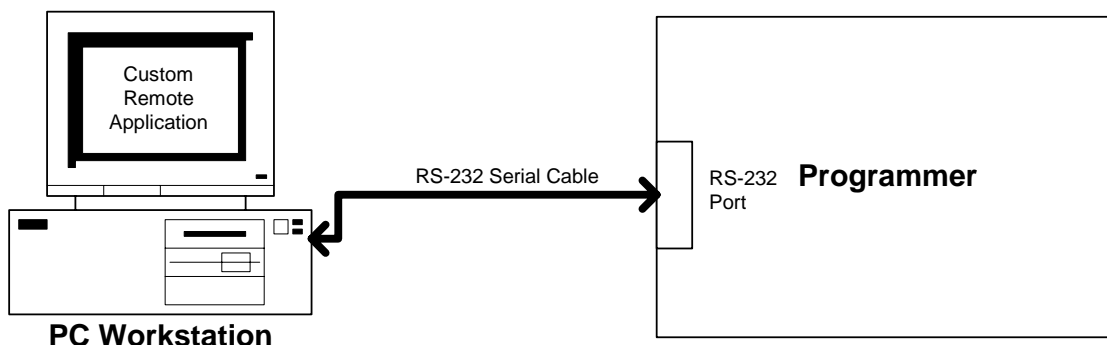
Programmer	EVAL version	STANDARD Development Pack	STANDARD Run-time License	PROFESSIONAL (PRO) Development Pack	PROFESSIONAL (PRO) Run-time License
EPSILON5	N/A	N/A	N/A	N/A	N/A
FS2000A	N/A	FS2000A-UPG10S	FS2000A-UPG10SR	N/A	N/A
FS2003	N/A	FS2003-UPG10S	FS2003-UPG10SR	N/A	N/A
FS2009	N/A	FS2009-UPG10S	FS2009-UPG10SR	N/A	N/A
PPM3-MK2	N/A	PPM3A1-UPG10S	PPM3A1-UPG10SR	PPM3A1-UPG10M	PPM3A1-UPG10AR
PPM4-MK1	Enabled	PPM4-UPG10S	PPM4-UPG10SR	PPM4-UPG10M	PPM4-UPG10AR
ISPnano Series I/II/III	Enabled	ISPnano-UPG10S	ISPnano-UPG10SR	ISPnano-UPG10M	ISPnano-UPG10AR

Please note:

1. A chargeable '**License upgrade**' is required for all programmers to enable them for either the '**STANDARD**' or '**PRO**' version

1.7 PC control of programmer via RS-232

An Equinox programmer can be remote controlled from a so-called 'Remote Application' running on a PC Workstation as detailed in the diagram below.



The '**Remote Application**' is a custom application which sends the required commands via ConsoleEDS to the programmer via the RS-232 link.

1.8 Device (Target IC) programming support

The '**ConsoleEDS**' protocol is, as far as possible, independent of the Target Device (Target IC) which is being programmed. The majority of **ConsoleEDS** commands are generic and will therefore work for all Target IC's whether it be a Serial EEPROM, Serial DataFLASH or a complex microcontroller with FLASH, EEPROM, Configuration Fuses and Lock Bits. This makes it very straightforward to change between different Target IC's as the control software does not have to change.

Please note:

There are a few '**device specific**' commands which are used to support specific features of a particular IC.

When creating a sequence of **ConsoleEDS** commands for a particular IC, it is important to understand how that IC will respond to these commands. Each device family will have a different organisation of memory and the Configuration Fuses and Lock Bits may operate in a different manner. It is highly recommended that you read the relevant application note for the IC which you are trying to program before attempting to program it with **ConsoleEDS**.

The table below lists the Application Notes available for helping to create '**Programming Projects**' for different device families.

Application Note	Device Family	Programming Interface
AN100	Atmel - AT89Sxxxx FLASH microcontrollers	SPI
AN101	Atmel - AVR FLASH microcontrollers	SPI
AN105	Atmel - AVR FLASH microcontrollers	JTAG
AN118	Generic I2C 24xxx Serial EEPROM memories	I2C
AN122	Atmel - AT91SAM7 ARM7 FLASH microcontrollers	JTAG
AN127	Atmel - ATxmega AVR FLASH microcontrollers	PDI / JTAG
AN128	NXP – LPCxxx ARM7 FLASH microcontrollers	JTAG

The latest versions of these application notes can be downloaded from the Equinox website.

1.9 Typical ConsoleEDS control example

ConsoleEDS is an ideal solution for programming both simple memory devices and complex ICs such as modern microcontrollers. Many programmable microcontrollers feature FLASH memory, EEPROM memory, Configuration fuses and Security Fuses / Lock Bits. **ConsoleEDS** supports programming of all of the different memory areas using very simple commands.

Example:

A typical example is to program an Intel HEX file called '**FlashFile.hex**' into the FLASH area of the target microcontroller. This would be implemented as follows:

ConsoleEDS BaseProject.prj /FLASHWRITE=FlashFile.hex

Where:

The **BaseProject.prj** is used by **ConsoleEDS** to define the Target IC, Target Voltage / Current, programming interface and other programmer settings.

The table below describes the most commonly used **ConsoleEDS** commands:

Programming action	ConsoleEDS example command(s)
To program a HEX or Binary file into the FLASH area of a device	/FLASHWRITE=FlashFile.hex
To program a HEX or Binary file into the EEPROM area of a device	/EEPROMWRITE=EEPROMFile.hex
To program the ' Configuration Fuses ' of a target microcontroller	/FUSEWRITE=0x01,0x02,0x03
To program the ' Security Fuses ' of a target microcontroller	/WRITESECURITY=0x03
To execute a pre-uploaded ' Standalone Programming Project ' on a selected programmer	/AUTOPROGRAM=ProjectName

1.10 Using ConsoleEDS to program unique Serial Numbers

ConsoleEDS is an ideal solution for programming a unique '**Serial number**' or a '**MAC Address**' into the FLASH or EEPROM area of a target IC.

The unique serialised data can be generated using any of the methods detailed below:

- A third party application generates the unique data and then passes it to ConsoleEDS as a file
- ConsoleEDS uses the built-in '**Incremental repository**' utility to generate the unique data for each Target IC.

Example:

An 8-byte incremental '**Serial number**' is to be programmed into the FLASH area of a target IC. The '**Serial number**' is to be generated by an external application and then passed to ConsoleEDS in a binary file called **SerialNumber.bin**. The start address of the '**Serial number**' in FLASH is to be **0x0200**.

The following command would be used to program this '**Serial number**' into the target IC:

ConsoleEDS BaseProject.prj /FLASHWRITE=SerialNumber.bin,0x0200,0x207,0

Where:

The **BaseProject.prj** is used by **ConsoleEDS** to define the Target IC, Target Voltage / Current, programming interface and other programmer settings.

Please refer to the separate application note AN129 which features many different examples of how to implement '**Serial number**' and '**MAC Address**' programming using ConsoleEDS.

1.11 Using ConsoleEDS to program unique Calibration Data

ConsoleEDS is an ideal solution for programming unique '**Calibration Data**' into or reading '**Calibration Data**' back from the FLASH or EEPROM area of a target IC. The '**Calibration Data**' could be A/D calibration coefficients, oscillator trim settings, power supply trim setting or any other data which varies from target board to target board.

Example:

A block of 32 bytes of '**Calibration Data**' is to be read from the EEPROM area of a target microcontroller starting at address 0x000 and is then to be re-programmed back into the FLASH area of the same microcontroller starting at address 0x03000.

The following command would be used to read this '**Calibration Data**' from the EEPROM area of the target IC:

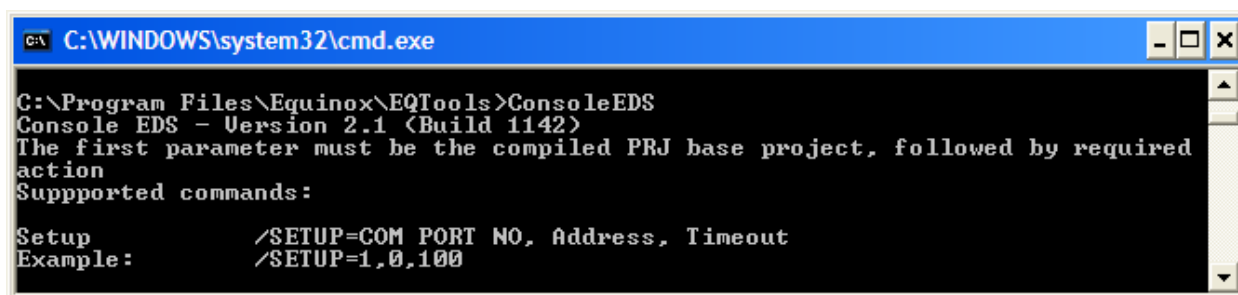
ConsoleEDS BaseProject.prj /FLASHWRITE=CalibrationData.bin,0x3000,0x301F,0

The following command would be used to program this '**Calibration Data**' into the FLASH area of the target IC starting at address **0x3000**:

ConsoleEDS BaseProject.prj /FLASHWRITE=CalibrationData.bin,0x3000,0x301F,0

1.12 Using ConsoleEDS from the Command Window

- The simplest way to test and become familiar with **ConsoleEDS** is to experiment using it from within a '**Command Window**' in Windows. This is as simple as launching a '**Command Window**' and then simply typing any ConsoleEDS command.
- Type '**ConsoleEDS**' → ConsoleEDS will launch and list all the arguments which can be used with ConsoleEDS



```
C:\WINDOWS\system32\cmd.exe

C:\Program Files\Equinox\EQTools>ConsoleEDS
Console EDS - Version 2.1 (Build 1142)
The first parameter must be the compiled PRJ base project, followed by required
action
Supported commands:

Setup           /SETUP=COM PORT NO, Address, Timeout
Example:        /SETUP=1,0,100
```

1.13 Using ConsoleEDS from a batch file

Once you have experimented with **ConsoleEDS** by manually typing commands in a '**Command Window**', it is then straightforward to make a list of these commands in a so called '**batch file**' which can then be executed sequentially.

Example:

The example below will program a target microcontroller FLASH, EEPROM, Configuration Fuses and Security Lock Bits by executing each command in sequence:

```
ConsoleEDS BaseProject.prj /FLASHWRITE=FlashFile.hex
ConsoleEDS BaseProject.prj /EEPROMWRITE=EEPROMFile.hex
ConsoleEDS BaseProject.prj /FUSEWRITE=0x01,0x02,0x03
ConsoleEDS BaseProject.prj /FLASHWRITE=/WRITESECURITY=0x03
```

Creating a batch file:

This sequence of commands can be saved as a simple text file with a suitable file name and file attribute of ***.bat**. Let's say we call the batch file **ProgramChip.bat** for this example. The batch file can then be called / executed from any other application running in Windows by simply specifying it by the name **ProgramChip.bat**.

Advantages of using batch files:

- Simple to create as no knowledge of any programming language is required
- Simple to test – just double-click the batch file in Windows
- No external control application is required
- Completely independent of any control application allowing the programming section to be tested independently of the control application
- Simple to version control and up-keep in a production environment
- Launching batch files is possible from most external applications eg. Visual Basic, C++, Labview, Labwindws CVI etc.

2.0 ConsoleEDS product versions

2.1 Overview

ConsoleEDS is available in three different versions as follows:

- **ConsoleEDS EVAL** (Evaluation)
- **ConsoleEDS STANDARD**
- **ConsoleEDS PRO** (Professional)

Each version caters for a different type of end-user application.

- The **'EVAL'** version is designed for evaluating ConsoleEDS and can also be used for programming a Target IC in the **'Code Development Environment'**. It has been restricted so that it can only program the FLASH area of a Target IC from a file.
- The **'STANDARD'** version has been designed for customers looking to control a single programmer from a **Remote Application**.
- The **'PROFESSIONAL'** version supports control up to 32 programmers from a single ConsoleEDS and comes with full scripting support. This allows very complex programming sequences to be created.

2.2 Development Pack and Run-time license

ConsoleEDS is also available as **"Development Pack"** or a **"Run-time License"**.

The **"Development Pack"** is required in order to get started with ConsoleEDS. It includes all the relevant licenses for either ConsoleEDS **'STANDARD'** or **'PRO'** plus documentation and technical support from Equinox to get your first ConsoleEDS project up-and-running.

The **"Run-time License"** is required for each additional programmer after the initial **"Development Pack"** has been purchased. It includes all the relevant licenses for either ConsoleEDS **'STANDARD'** or **'PROFESSIONAL'** for a single programmer. No documentation or technical support is offered with this version. It is aimed at customers who are already proficient at using ConsoleEDS and who have already purchased a **"Development Pack"**.

Please note:

A **"Development Pack"** must be purchased for the first programmer with which ConsoleEDS is to be used. Each subsequent programmer after the first programmer only requires a **"Run-time License"**.

2.3 Comparison of ConsoleEDS versions

The table below details the functionality of the different versions of **ConsoleEDS**:

ConsoleEDS version	Evaluation	Standard	Professional
Intended use	Firmware Development	Production use with a single programmer	Production use with single or multiple programmers
Technical support	None	e-mail only	phone and e-mail
Number of programmers which can be controlled from a single session of ConsoleEDS	1 programmer	1 programmer only	<ul style="list-style-type: none"> Multiple programmers simultaneously Up to 16 x PPM3-MK2 programmers Up to 32 PPM4-MK1 or ISPnano programmers
Uploading 'Standalone Programming Projects' to programmer(s) using /UPLOAD command	No	1 programmer only	<ul style="list-style-type: none"> Multiple programmers simultaneously Up to 16 x PPM3-MK2 programmers Up to 32 x PPM4-MK1 or ISPnano programmers
Chip Erase Operation	Target Chip is erased every time	Erase operation is user selectable	Erase operation is user selectable
Programming of FLASH area	YES Restricted so address range is not selectable	YES – File offset and Target Chip address range are user selectable	YES – File offset and Target Chip address range are user selectable
Programming of EEPROM area	Not supported	YES – File offset and Target Chip address range are user selectable	YES – File offset and Target Chip address range are user selectable
Programming of Configuration Fuses	Not supported	YES	YES
Programming of Security Fuses	Not supported	YES	YES
Measurement of Target Voltage(s) and Current	Not supported	YES	YES
Execution of programming scripts / scripting commands	Not supported	Not supported	YES
Automated Serial Number / MAC address programming	Not supported	Not supported	YES

2.4 Software Utilities included in each version of ConsoleEDS

The table below details which utilities are included with the Evaluation, Standard and Professional versions of ConsoleEDS.

ConsoleEDS version	Evaluation (EVAL)	Standard (STANDARD)	Professional (PRO)
Standalone Project Upload Wizard	No	YES 1 programmer only	YES Up to 16 programmers using PPM3-MK2 Up to 32 programmers using PPM4-MK1 and ISPhano
Script Creation / Compilation	No	No	YES (DEVELOPMENT Pack only)
Script Execution	No	No	YES
ConsoleEDS Multi-port	No	No	YES
ConsoleEDS Multi-node	No	No	YES

2.5 ConsoleEDS - EVAL

The **EVALUATION** version of ConsoleEDS is designed to allow a firmware developer to integrate the Equinox ISP Programming functionality into the IDE (Integrated Development Environment) which is being used for code development. ConsoleEDS can then be used to automatically program the output file(s) (HEX or Binary) from any Compiler or IDE into the programmable memory area of the Target IC on a Development Board or Customer Application Board.

This version is restricted to writing a HEX or BINARY file from disk into the Target IC.

It is not possible to program Configuration or Security Fuses with this version.

It is also only possible to program the FLASH area of a Target IC. Programming of the EEPROM area is not supported.

ConsoleEDS EVAL version is now bundled with EQTools.

A typical example of using the EVAL version of ConsoleEDS to program a file into a Target IC is as follows:

ConsoleEDS BaseProject.prj /FLASHWRITE=FileName.hex

Where:

- **BaseProject.prj** is a project file on your hard disk containing the settings for this session of ConsoleEDS i.e. which programmer, chip, voltage etc.
- **FileName.bin** is the file name of a BINARY data file
- **FileName.hex** is the file name of an Intel Hex data file

2.6 ConsoleEDS STANDARD

The STANDARD version of ConsoleEDS offers full control of the attached Equinox Programmer via a comprehensive set of Command Line commands. It is possible to not only program data from a file, but also read back blocks of data to file, program / read Configuration / Security Fuses etc. It is also possible to upload 'Standalone Programming Projects' to an attached programmer from the Command Line.

ConsoleEDS STANDARD Features:

- Supports writing of a block of data from a file to the FLASH or EEPROM of a Target Chip
- Supports reading of a block of data from the FLASH or EEPROM of the Target Chip to a file on the PC hard disk
- Supports uploading of pre-compiled standalone '**Standalone Programming Projects**' to a single programmer using the **/UPLOAD** command without requiring EQTools or the Project Upload Wizard.
- Supports running of a specified 'Standalone Programming Project' which has already been uploaded into the programmer 'FLASH Memory Store'
- Supports programming and reading back of the 'Configuration Fuses' (fuse bits) of the Target IC
- Supports programming and reading back of the 'Security Fuses' of the Target IC
- Supports reading the Device ID (Signature) of the Target IC
- Supports reading / writing of Atmel AVR on-chip Oscillator Calibration Byte

It is necessary to purchase a '**License Upgrade**' in order to run **ConsoleEDS Standard**. For ordering information, please see section 1.6.

2.7 ConsoleEDS - PRO

The 'PRO' version of ConsoleEDS supports all the same features as ConsoleEDS STANDARD.

It also supports the following additional features:

- ConsoleEDS Multi-Port (supports operation of multiple sessions of ConsoleEDS on multiple PC Serial ports)
- ConsoleEDS Multi-Node (supports simultaneous control of up to 32 programmers)
- Scripting Commands (supports executing of 'Programming Scripts')
- Supports automatic generation of an incremental serial number or MAC address which can then be programmed into the FLASH or EEPROM area of the Target Chip.

It is necessary to purchase a '**License Upgrade**' in order to run **ConsoleEDS PRO**. For ordering information, please see section 1.6.

2.8 ConsoleEDS Multi-Port

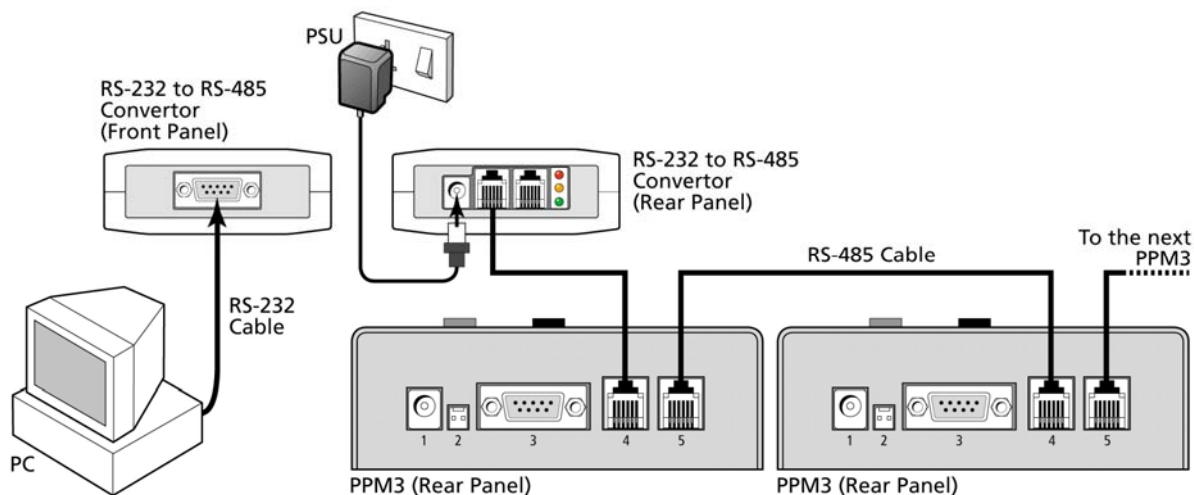
The STANDARD version of ConsoleEDS supports controlling of a single Equinox Programmer on a selected PC COM port. If 'ConsoleEDS Multi-Port' is enabled, then it is possible to set up a PC with multiple COM ports and then connect a programmer to each COM port. ConsoleEDS can then be used to control each programmer in turn by specifying which COM port to use on the ConsoleEDS command line.

This technique allows a single PC to control multiple programmers (egg. 2, 4, 8) which is ideal for high-volume manufacturing / test environments.

2.9 ConsoleEDS Multi-Node

The PRO version of ConsoleEDS supports controlling of multiple Equinox Programmers via a single ConsoleEDS session. This allows a single ConsoleEDS session to control up to **16 x PPM3-MK2** or **32 x PPM4-MK1** or **32 x ISPnano** programmers which are networked together on an RS485 network to a single PC COM port.

A typical setup with two networked PPM3-MK2 programmers is shown in the figure below:



It is also possible to control multiple programmers at the same time using a single ConsoleEDS session. In this case the **/NODES** command is used. In the example below, ConsoleEDS will send the same command to 3 programmers at the same time at addresses 0, 1 and 2.

Example:

ConsoleEDS BaseProject.prj /AUTOPROGRAM=Project1 /NODES=3,0,1,2 /TOTALTIMEOUT=60

This allows ConsoleEDS to start a programming operation on multiple programmers at the same time. It then automatically polls each specified programmer until they have all finished the execution of the specified command.

Please note:

- The '**Multi-Node**' functionality was introduced in **ConsoleEDS** build 971.
- It is only available with the '**PRO**' version of **ConsoleEDS**.

2.10 Which version of ConsoleEDS is my programmer enabled for?

All Equinox ISP Programmers are enabled for the '**EVALUATION**' version of ConsoleEDS.

To check whether your programmer is ENABLED for '**ConsoleEDS STANDARD**' or '**ConsoleEDS PRO**', please follow the steps below:

- Connect the programmer to a PC spare COM port and apply power to the programmer.
- For the PPM3-MK2, PPM4-MK1 and ISPnano programmers, make sure that the '**Communication Node Address**' is set to address '**0**' and that '**RS-232 Mode**' is selected.
- Launch EQTools
- From the top menu bar, select **<Programmer><Programmer Info>**
- EQTools will now communicate with your programmer and request the 'programmer settings'
- From the displayed list of options
- From the list of options, please check whether '**ConsoleEDS-STANDARD**' or '**ConsoleEDS-PROFESSIONAL**' is ENABLED
- If you plan to have multiple Equinox programmers connected to multiple PC COM ports, check that the option '**ConsoleEDS-Multiport**' is also ENABLED.

If the required version of ConsoleEDS is DISABLED, then your programmer will require a '**License Upgrade**' to allow this feature. Please contact Equinox via e-mail at sales@equinox-tech.com to discuss your requirements further.

2.0 ConsoleEDS Installation Instructions

2.1 Overview

The ConsoleEDS application is installed as standard when installing the EQTools application. The ConsoleEDS executable can be found in the **`\program files\equinox\eqtools`** directory. It is entirely possible to move the ConsoleEDS application to any other file location as it does not rely on EQTools for its operation. It is also possible to run ConsoleEDS as a 'Standalone Application' without even installing EQTools as long as your application is not using the Equinox Interface Database to store the Project List or to pass parameters to/from ConsoleEDS.

To install ConsoleEDS, please locate and run the EQTools installation program. This will install EQTools, ConsoleEDS and the Upload Wizard utilities onto your PC.

2.2 Installation instructions

The ConsoleEDS application is installed as standard when installing the EQTools application.

Instructions:

- Locate the latest version of the '**EQTools**' software suite on the Equinox website. (click the **<Downloads>** tab and search for '**EQTools**').
- Run the EQTools installation executable by double-clicking it
- Follow the on-screen prompts during the installation until you get to the 'Install type' screen.
- If you wish to install EQTools and ConsoleEDS, simply select all the default options.
- If you wish to only install ConsoleEDS, select the '**Custom setup**' and then select to only install ConsoleEDS.

Once the EQTools installation is complete, the ConsoleEDS executable can be found in the **`\program files\equinox\eqtools`** directory. The location of all other files is detailed in the next section.

Please note:

- It is possible to move the ConsoleEDS.exe application to a different directory.
- This can make it simpler to construct the command line commands as the path description to ConsoleEDS can then be reduced.
- e.g. **`c:\program files\equinox\eqtools\ConsoleEDS /PROGINFO`** could be reduced to e.g. **`c:\test\ConsoleEDS /PROGINFO`**.

2.3 Installation Directory Structure

The default installation directory for EQTools and ConsoleEDS is: **\\program files\\equinox**

The installation creates the folders listed in the table below:

Main Directory	Sub Directory	Description
\\eqtools		Main applications + associated files <ul style="list-style-type: none"> • EQTools.exe • ConsoleEDS.exe • Default.rst
	\\lib	Device Library Files (*.lib)
	\\examples	Example projects / scripts
\\db		<ul style="list-style-type: none"> • Equinox Interface Database Access 2000 Database – filename: ac_isppro.mdb • ADOExplorer – Database utility
\\UploadWizard		Upload Wizard application files <ul style="list-style-type: none"> • UploadWizard.exe
\\Firmware		This directory contains the latest firmware update files for each Equinox programmer.

2.4 Running ConsoleEDS as a Standalone Application

ConsoleEDS does not rely on any other applications in order to run. It does not even need EQTools installed on the PC where it is running as all commands are self-contained. The EQTools application will be required to create and maintain the Programming Projects, Fuse & Security files etc. However, once all the relevant files have been created and tested, it is then possible to use all these files within ConsoleEDS completely independently of EQTools.

The only exceptions to this are as follows:

1. When using the ConsoleEDS command for programming incremental numbers:

eg. *ConsoleEDS ProjectName.prj /SETUPINC*

This command uses the **Incremental Repository** in EQTools and so requires that EQTools is installed on the same PC which is running ConsoleEDS.

2. When using Programming Scripts

eg. *ConsoleEDS ProjectName.prj /SCRIPT=ScriptName.esf*

This command executes pre-compiled Programming Scripts. If the script uses the Interface Database then it is necessary to run either the EQTools or ISP-PRO installation in order to install the database.

2.5 Moving ConsoleEDS application to your “Working Directory”

When you are controlling ConsoleEDS from a Remote Application, it is **not recommended** to place your working project files in the ***\\program files\\equinox\\eqtools*** directory.

The reasons for this are as follows:

- If someone installs a newer version of EQTools onto the PC, this installation may overwrite some of your ‘Project Files’ by mistake.
- There may be file permission problems on certain PC’s when accessing the ***\\program files\\equinox\\eqtools*** directory.

It is possible to move the **ConsoleEDS.exe** application to a different directory on your PC e.g. your **“working directory”** where all your “Project Files” are stored. Moving ConsoleEDS can also make it simpler to construct the command line commands as the path description to ConsoleEDS can then be simplified.

Example:

c:\\program files\\equinox\\eqtools\\ConsoleEDS /PROGINFO

could be simplified to

c:\\test\\ConsoleEDS /PROGINFO.

3.0 ConsoleEDS – Getting Started Guide

3.1 Overview

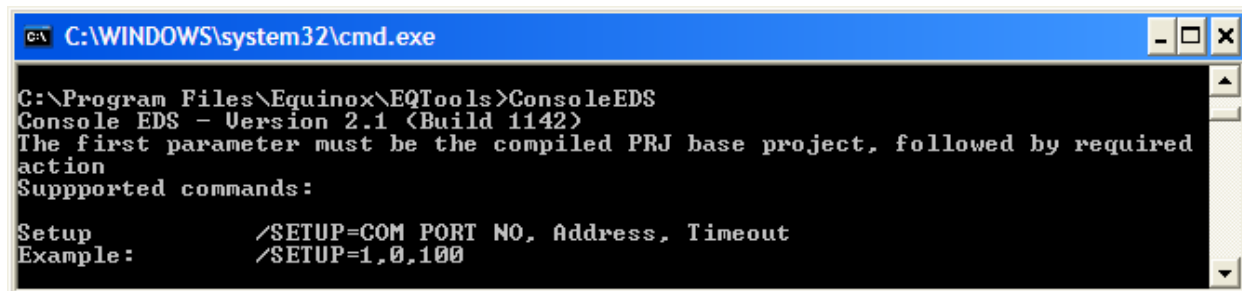
ConsoleEDS is '**Console Application**' which allows any Equinox ISP Programmer to be controlled via simple '**Command Line**' commands from other applications running under Windows. The simplest way to experiment with **ConsoleEDS** is by typing commands in a Windows '**Command Window**'. This procedure is described in the section 3.2 onwards. It is also possible to control ConsoleEDS from a remote Visual Basic, C++ or Labview application. This is beyond the scope of this introductory guide, but the explanations here should provide most of the information required to implement this support.

3.2 Testing ConsoleEDS installation

The simplest way to test and become familiar with **ConsoleEDS** is to experiment using it from within a '**Command Window**'.

To open a '**Command Window**':

- Select **<Start><Run>**
- Type '**cmd**' in the '**Run**' box → a **Command Window** will open
- Select the **Command Window** by clicking anywhere in the window
- At the prompt, type '**cd program files\equinox\eqtools**' → you should now be in the **EQTools** directory.
- Type '**dir *.exe**' → you should see the file **ConsoleEDS.exe**
- Type '**ConsoleEDS**' → ConsoleEDS will launch and list all the arguments which can be used with ConsoleEDS



```
C:\WINDOWS\system32\cmd.exe

C:\Program Files\Equinox\EQTools>ConsoleEDS
Console EDS - Version 2.1 (Build 1142)
The first parameter must be the compiled PRJ base project, followed by required
action
Supported commands:

Setup          /SETUP=COM PORT NO, Address, Timeout
Example:       /SETUP=1,0,100
```

3.3 Setting the Communications Parameters

ConsoleEDS works by sending serial commands via one of the PC COM Ports to an attached Equinox Programmer. It is therefore necessary to set up ConsoleEDS initially to tell it which COM port to use, which communications address the programmer is on and the 'Communications Timeout Delay'. If you have already installed and used EQTools, then ConsoleEDS will use the defaults already set up in the last session of EQTools.

The command to set up the Communications Parameters is as follows:

Setup ConsoleEDS /SETUP=COM PORT Number, Programmer Node Address, Timeout

Example: ConsoleEDS /SETUP=1,0,1000 or ConsoleEDS "/SETUP=1,0,1000"

Where:

- **COM PORT** – is the physical PC COM Port number – defaults to COM1
- **Address** – is the 'Communications Node Address' of the attached programmer. This should be set to '0' for all programmers by default. If you are using a PPM3-MK2 programmer in RS485 mode, then you may need to set the address to a different value.
- **Timeout** – this is the amount of time allowed from sending a ConsoleEDS command to generating a 'Communications Timeout Error'. The default timeout delay is 1 second (1,000 ms).
- Please make sure there are no spaces between the parameters.

ConsoleEDS will respond as follows:

Console EDS - Version 2.1 (Build 1150)

Setup options updated to be:

COM Port: 1

Baud rate: 38400

Node address: 0

Timeout(ms): 1000

Closing Console EDS, no errors have occurred.

Please note:

ConsoleEDS will use the same COM port as specified by EQTools if the ConsoleEDS **/SETUP** command is not used.

3.4 Reading the Programmer Information

It is possible to read back the Programmer Information from the attached programmer using ConsoleEDS. The information returned includes the Programmer Serial Number, Programmer Type etc and all the optional License Upgrade information.

To read the Programmer Information back from the attached programmer, type the following command at the prompt in the 'Command Window':

ConsoleEDS /PROGINFO

ConsoleEDS will respond as follows:

Console EDS - Version 2.1 (Build 730)

Programmer Information:

Programmer details for attached device using: COM Port:1

Baud rate:38400, Node address:0

Serial Number - 1028

Programmer Type - 25 - PPM3 MK2

Firmware - Version 2.54m

Micro Revision - G

Hardware Version - 1.6

Update Date - 19/4/2006

Test Date - 27/6/2005

Number of Updates - 1

Programmer Options: -

Equinox Development Suite (EDS) - Enabled

ASCII Text Communications Protocol. - Enabled

ISP-PRO Script Execution - Disabled

ISP-PRO Script Compilation - Disabled

ISP-PRO Script Text Editor - Disabled

ISP-PRO Remote Application Control - Disabled

ISP-PRO Labview Control - Disabled

Communications Debug - Disabled

Programmer Credits - Disabled

Atmel ATmega JTAG ISP - Disabled

ConsoleEDS-PRO - Disabled

ConsoleEDS-MULTIPOINT - Disabled

Standalone Upload Utility - Disabled

Closing Console EDS, no errors have occurred.

3.5 ConsoleEDS Command Format

The ConsoleEDS Command Format for any command which interacts with a Target IC is as follows:

ConsoleEDS "BaseProject.prj" "/argument1" "/argument2"

Here is a simple example of using ConsoleEDS to program a file into the FLASH and EEPROM areas of a Target IC:

ConsoleEDS "BaseProject.PRJ" "/FLASHWRITE=Ffile.bin" "/EEPROMWRITE=Efile.bin"

#	Command	Description
	BaseProject.prj	This is a compiled 'Programming Project' which describes the Target Chip, Programmer, SPI / JTAG speeds, Target Voltage etc.
	/FLASHWRITE	This command programs the specified file into the Target IC FLASH area.
	/EEPROMWRITE	This command programs the specified file into the Target IC EEPROM area.

The '**BaseProject.prj**' is required by ConsoleEDS so that it knows which IC is being programmed, what the SPI / JTAG frequencies are, what the Target Voltage is etc.

Please note:

It is a good idea to put double **quote marks** (" ") around each argument declared on the ConsoleEDS Command Line. This contains the argument and makes sure that ConsoleEDS does not misinterpret certain characters. It also allows you to use space characters in file names.

3.6 Reading the 'Device ID (Signature)' of a Target IC

It is often a good idea to make sure that you can read the 'Device ID (Signature)' of a Target IC using ConsoleEDS before attempting any more complicated operations. This will make sure that your 'Base Project' settings are correct and that the Target IC is definitely responding to the programmer.

You will need to create a Programming Project (*.prj) which has the Target IC selected.

This is called the '**BaseProject**' and is used by ConsoleEDS to define all the parameters used for programming. You can create the '**BaseProject**' using EQTools – EDS or Project Builder.

To read the Device ID (Signature) from the Target IC, type the following command:

ConsoleEDS "BaseProject.PRJ" /READSIG

ConsoleEDS will respond as follows:

Console EDS - Version 2.1 (Build 730)

Console EDS - Reading signature

Console EDS - Signature read back: 0x1E9007

The Device ID of '**0x1E9007**' is for an Atmel ATtiny13 IC. The signature will be different depending on the Target IC selected.

4.0 ConsoleEDS Command Specification

4.1 Overview

ConsoleEDS is a Console Application running under Windows. It allows programming actions to be performed by issuing simple commands via the Command Line. This section details the available command set for ConsoleEDS and also a short explanation of how to use each command.

4.2 ConsoleEDS Command Format examples

All ConsoleEDS sessions must begin with the '**ConsoleEDS**' command followed by any other commands and/or arguments. It is a good idea to put double quote marks around each command to make sure that ConsoleEDS does not misinterpret any of the characters in the arguments.

Example 1

ConsoleEDS "/SETUP=1,0,1000"

→ sets up the communication parameters for this session of ConsoleEDS.

Example 2

The ConsoleEDS Command Format for any command which interacts with a Target IC is as follows:

ConsoleEDS "BaseProject.prj" "/argument1" "/argument2"

Here is a simple example of using ConsoleEDS to program a file into the FLASH and EEPROM areas of a Target IC:

ConsoleEDS "BaseProject.PRJ" "/FLASHWRITE=Ffile.bin" "/EEPROMWRITE=Efile.bin"

#	Command	Description
	BaseProject.prj	This is a compiled 'Programming Project' which describes the Target IC, Programmer, SPI / JTAG speeds, Target Voltage etc.
	/FLASHWRITE	This command programs the specified file into the Target IC FLASH area.
	/EEPROMWRITE	This command programs the specified file into the Target IC EEPROM area.

The '**BaseProject.prj**' is required by ConsoleEDS so that it know which IC is being programmed, what the SPI / JTAG frequencies are, what the Target Voltage is etc.

Please note:

- Commands can be in small or CAPITAL letters.
- A **<CR>** (carriage return character) must be sent after each command.
- It is a good idea to put double **quote marks (" ")** around each argument declared on the ConsoleEDS Command Line. This contains the argument and makes sure that ConsoleEDS does not misinterpret certain characters.

4.3 Programmer Setup Commands

4.3.1 Overview

The commands in this section are used to set up ConsoleEDS to control one or more programmers connected to a PC. The **/SETUP** command is used to specify the PC COM port to be used including setting up the correct COM port, checking ConsoleEDS version and checking the programmer firmware version. These commands usually need to be executed only once at the start of a production run.

ConsoleEDS command	Description
/SETUP	Sets up the ConsoleEDS communications parameters
/COMPORT	Sets up the PC COM Port (use /SETUP instead)
/BAUDRATE	Sets the 'Communications BAUD Rate' between the PC and the attached programmer(s).
/PROGINFO	Returns the 'Programmer Information' (serial number, firmware version etc)
/CHECKMINIMUMBUILD	Checks that the installed version of ConsoleEDS is >= to the specified version.
/CHECKMINIMUMFIRMWARE	Checks that the attached programmer is running the specified version of firmware

4.3.2 /SETUP command

This command specifies the ConsoleEDS communications parameters which will be used to communicate with the attached programmer(s).

PC command sent	
ConsoleEDS command	/SETUP
Command format	ConsoleEDS /SETUP=COM PORT Number, Node Address, Timeout Example: /SETUP=1,0,1000
Command arguments	<ul style="list-style-type: none"> • COM PORT NUMBER - Is the COM Port (PC Serial Port) to which the programmer is attached • Address - is the communications address of the programmer. This is always '0' for all programmers except for the PPM3-MK2 where it can be 0 – 15 depending on the address set up on the DIP switches. • Timeout - This is the delay after sending a command to the programmer before ConsoleEDS reports a Timeout Error. The default timeout period is 1 second. (1,000 ms)
ConsoleEDS debug output	ConsoleEDS /SETUP=1,0,1000 Setup options updated to be: COM Port: 1 Baud rate: 38400 Node address: 0 Timeout(ms): 1000 Result: PASS Closing Console EDS, no errors have occurred.

Please note:

- If the **/SETUP** command is not used to set up the communications parameters then ConsoleEDS will use the default 'Communications Settings' from EQTools (if installed on the PC).
- Please make sure there are no spaces in between the parameters.

4.3.3 /COMPORT command

This command specifies which PC COM port (Serial Port) ConsoleEDS will use to communicate with the attached programmer(s).

PC command sent	
ConsoleEDS command	/COMPORT
Command format	ConsoleEDS /COMPORT=1 (selects COM1 - DEFAULT) ConsoleEDS /COMPORT=2 (selects COM2)
Command arguments	<ul style="list-style-type: none"> COM PORT NO - Is the COM Port (PC Serial Port) to which the programmer is attached

Please note:

- It is better to use the **/SETUP** command to set up the communications parameters for ConsoleEDS.
- The default COM port is COM1.
- If you are using the default COM port, then you do not need to specify the **/COMPORT** parameter.

4.3.4 /BAUDRATE command

This command specifies the communications speed (BAUD rate) which will be used to communicate with the attached programmer(s).

PC command sent	
ConsoleEDS command	/BAUDRATE
Command format	ConsoleEDS /BAUDRATE=baudrate
Command arguments	<ul style="list-style-type: none"> baudrate – This is the baudrate to which the programmer is to be switched to.

Please note:

- A programmer will always reset to 38,400 baud rate after power-up.
- The **/BAUDRATE** command should then be used to set the baud rate to a faster speed.
- If ConsoleEDS cannot communicate with the attached programmer at the faster baud rate, it will automatically try again at 38,400 baud.

4.3.5 /PROGINFO command

This command returns the serial number, firmware version etc. of an attached programmer

PC command sent	
ConsoleEDS command	/PROGINFO
Command format	ConsoleEDS /PROGINFO
Command arguments	None
ConsoleEDS debug output	CODE VERSION 2.29 SERIAL NUMBER 2050 HARDWARE VERSION 1.41 UPDATE DATE 28-6-2001

Please note:

- This command can only be used with a single programmer.
- Please use the **/DETECT** command if there are multiple programmers connected to the same COM port via RS485.

4.3.6 /CHECKMINIMUMBUILD command

This command checks that the version of ConsoleEDS installed on the PC is of the specified build version or above. This is a very useful housekeeping check to ensure that the ConsoleEDS version installed is compatible with the script being executed.

PC command sent	
ConsoleEDS command	/CHECKMINIMUMBUILD
Command format	ConsoleEDS /CHECKMINIMUMBUILD=BuildVersion
Command arguments	BuildVersion
ConsoleEDS debug output	ConsoleEDS /CHECKMINIMUMBUILD=903 Console EDS - Started at 30/05/2008 01:13:37 Console EDS - Checking ConsoleEDS Build number is >= 903 Console EDS - ConsoleEDS Build number is 923 Result: PASS Closing Console EDS, no errors have occurred.

4.3.7 /CHECKMINIMUMFIRMWARE command

This command checks that the version of firmware installed in the attached programmer is equal to or greater than the specified version. This is a very useful housekeeping check to ensure that the firmware version in the attached programmer(s) is definitely compatible with the instructions in the ConsoleEDS script.

PC command sent	
ConsoleEDS command	/CHECKMINIMUMFIRMWARE
Command format	<i>ConsoleEDS /CHECKMINIMUMFIRMWARE=FirmwareVersion</i>
Command arguments	<i>FirmwareVersion e.g. 3.04</i>
ConsoleEDS debug output	<i>ConsoleEDS /CHECKMINIMUMFIRMWARE=3.07</i> <i>Console EDS - Checking Programmer Firmware version is >= 3.07</i> <i>Console EDS - Programmer Firmware is version 3.07</i> <i>Result: PASS</i> <i>Closing Console EDS, no errors have occurred.</i>

Please note:

It is very important to make sure that the correct version of firmware is used when executing ConsoleEDS commands. This is because many ConsoleEDS commands and Programming Projects will only work properly on the version of firmware they were compiled for or a later version of firmware.

4.4 Using ConsoleEDS with multiple programmers

4.4.1 Overview of using multiple programmers

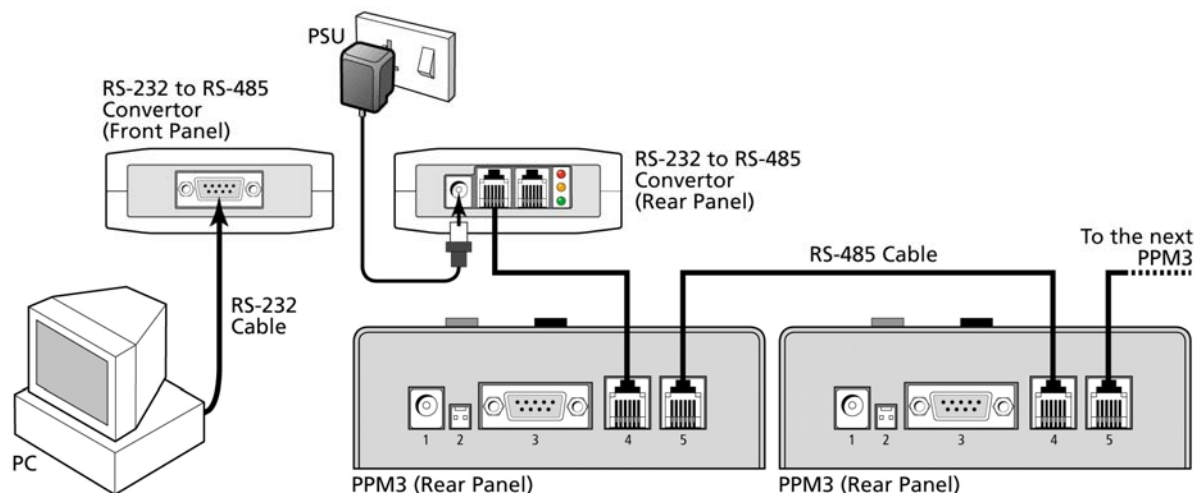
It is possible to network multiple programmers on an RS-485 network and control them from a single 'Supervisor PC' running a single session of ConsoleEDS. This scenario requires that each programmer is set to '**RS485**' operation and that a unique '**Communications Node Address**' is set for each programmer.

The table below details the programmers which support multi-node operation.

Programmer	Programmer channels	Communications node address range
PPM3-MK2	16	0...15
PPM4-MK1	16	0...15
ISPnano Series I/II	32	0...31
ISPnano Series III	32	0...31

An '**RS-232 to RS-485**' converter is also required to convert the RS-232 signal from the PC COM port to an optically isolated RS-485 signal suitable for interfacing with the programmers on the network.

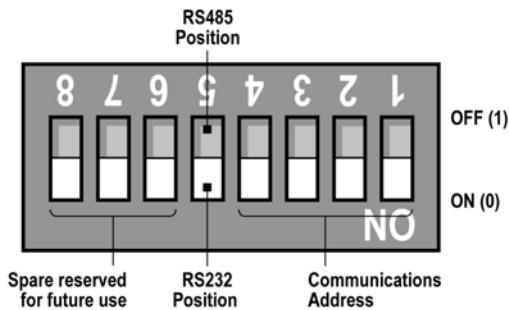
The typical setup for PPM3-MK2 / PPM4-MK1 programmers is shown in the figure below:



4.4.2 Setting the unique 'Node Address' on a PPM3-MK2 or PPM4-MK1

In order for ConsoleEDS to communicate with each programmer on the network, it is necessary to switch each programmer to '**RS485 Mode**' and then set up a unique '**Node Address**' for each programmer.

The layout of the DIP switch on the PPM3-MK2 / PPM4-MK1 programmer is shown in the diagram below.



- Select the '**RS485**' position
- Set up a unique '**Node Address**' in the range 0...15 using the right-most 4 DIP switches – see PPM3-MK2 User Manual.

4.4.3 How ConsoleEDS deals with multiple programmers

When controlling multiple programmers on an RS485 network using ConsoleEDS, it is necessary to think of each programmer as being at a unique '**Node Address**' in the range 0...15 for PPM-MK2 programmers and 0...31 for PPM4-MK1 and ISPnano programmers. If there is more than one programmer on the network, then the **/DETECT** command can be used to "detect" which programmers are connected to the RS485 bus and what addresses each programmer is on.

Example:

3 programmers have been detected:

Addr	Type	Firmware	Serial No
0	PPM3 MK2	3.07	500
1	PPM3 MK2	3.07	501
2	PPM3 MK2	3.07	502

It is possible to control an individual programmer at a specified address by simply specifying the **/NODE=NodeAddress** command. In the example below, ConsoleEDS will send the command to the programmer at "Address 3" only.

Example:

ConsoleEDS BaseProject.prj /AUTOPROGRAM=Project1 /NODE=3

It is also possible to control multiple programmers at the same time (concurrently) using a single ConsoleEDS command. In this case the **/NODES** command is used. In the example below, ConsoleEDS will send the same command to 3 programmers at the same time at addresses 0, 1 and 2.

Example:

ConsoleEDS BaseProject.prj /AUTOPROGRAM=Project1 /NODES=3,0,1,2 /TOTALTIMEOUT=60

This allows ConsoleEDS to start a programming operation on multiple programmers at the same time. There is a slight delay of eg. 100 – 200ms between starting programmers as ConsoleEDS actually has to start execution of each programmer in turn. Once all programmers have been started, ConsoleEDS then automatically polls each specified programmer until they have all finished the execution of the specified command.

The **/TOTALTIMEOUT** parameter is used to stop any ConsoleEDS command hanging indefinitely if a project gets stuck for some reason.

The table below lists the commands which are used to control multiple programmers on an RS485 network.

ConsoleEDS command	Description
/DETECT	Returns a list of all attached programmers. It is used to detect multiple PPM3-MK2, PPM4-MK1 or ISPhano programmers on an RS485 network.
/NODE	This command specifies which “Programmer node address” a ConsoleEDS command is sent to.
/NODES	This command enabled a ConsoleEDS command to be sent to multiple programmers on an RS485 network at the same time.
/AUTOPROGRAMBROADCAST	This is a special command which starts execution of a specified ‘standalone programming project’ at exactly the same instant on all programmers on the network.
/DELAYBEFOREPOLL	This command specifies the delay between starting execution of all programmers and when ConsoleEDS begins to poll each programmer to see if it has finished execution of the specified project.
/TOTALTIMEOUT	This command specifies the maximum amount of time which any ConsoleEDS session is allowed to execute for before timing out.

4.4.4 /DETECT command

This command is used to detect multiple PPM3-MK2, PPM4-MK1 or ISPnano programmers which are connected to the same COM port of the controlling PC on an RS485 network. It scans the specified COM port for any attached programmers and will scan from programmer '**Node address**' 0 to 31 looking for one or more programmers attached to this COM port.

PC command sent																	
ConsoleEDS command	/DETECT																
Command format	ConsoleEDS /DETECT																
Command arguments	None																
ConsoleEDS debug output	<p>ConsoleEDS /DETECT Console EDS - Version 2.1 (Build 971) Copyright 1998-2008 Equinox Technologies UK Limited Console EDS - Started at 12/10/2008 12:44:52 Now searching for programmers, please wait...</p> <p>Operation: Detect attached programmer(s) Result: PASS</p> <p>3 programmers have been detected:</p> <table><tr><th>Addr</th><th>Type</th><th>Firmware</th><th>Serial No</th></tr><tr><td>0</td><td>PPM3 MK2</td><td>3.07</td><td>500</td></tr><tr><td>1</td><td>PPM3 MK2</td><td>3.07</td><td>501</td></tr><tr><td>2</td><td>PPM3 MK2</td><td>3.07</td><td>502</td></tr></table> <p>Result: PASS Closing Console EDS, no errors have occurred.</p>	Addr	Type	Firmware	Serial No	0	PPM3 MK2	3.07	500	1	PPM3 MK2	3.07	501	2	PPM3 MK2	3.07	502
Addr	Type	Firmware	Serial No														
0	PPM3 MK2	3.07	500														
1	PPM3 MK2	3.07	501														
2	PPM3 MK2	3.07	502														

Please note:

The **/DETECT** command was introduced in ConsoleEDS build 971.

4.4.5 /NODE command

Each PPM3-MK2, PPM4-MK1 or ISPnano programmer connected on an RS485 network has a unique '**Node address**' (communications address). The **/NODE** command is used to specify which programmer in a multi-programmer system that a ConsoleEDS command should be sent to. It can be used with most ConsoleEDS commands.

PC command sent	
ConsoleEDS command	/NODE
Command format	ConsoleEDS /OtherCommands /NODE
Command arguments	None
ConsoleEDS debug output	ConsoleEDS BaseProject.prj /AUTOPROGRAM=Project1 /NODE=3

In the above example, ConsoleEDS will execute the /AUTOPROGRAM command on the programmer at address 3.

Please note:

The **/NODE** command was introduced in ConsoleEDS build 971.

4.4.6 /NODES command

It is possible to instruct more than one programmer on an RS485 network to perform a specified action using the /NODES command.

PC command sent	
ConsoleEDS command	/NODE
Command format	<i>ConsoleEDS /OtherCommands</i> <i>/NODES=NumberOfNodes,Address0,AddressN</i>
Command arguments	where: <ul style="list-style-type: none"> • NumberOfNodes is the number of programmers on the network. • Address0...AddressN is simply a list of the 'Node Addresses' of the programmers which the command is to be executed by.

Please note:

- ConsoleEDS starts the first programmer, checks that it has started and then starts the next programmer and so on. This means there may be a slight delay of eg. 100 – 200ms between the start of each programmer.
- If the result from all programmers is 'PASS', then ConsoleEDS will return ErrorCode=0
- If the individual result from any of the programmers is 'FAIL', then ConsoleEDS will return ErrorCode=1
- If you need all programmers to start at exactly the same time, then the **/AUTOPROGRAMBROADCAST** command should be used.
- If any of the programmers do not complete the specified command within the **"TOTALTIMEOUT"** period, then ConsoleEDS will quit with a **"Timeout Error"**.
- The **/NODES** command was introduced in ConsoleEDS build 971.

Example:

ConsoleEDS BaseProject.prj /AUTOPROGRAM=Project1 /NODES=3,0,1,2 /TOTALTIMEOUT=60

This command will execute the **"/AUTOPROGRAM=Project1"** command on 3 programmers at addresses 0, 1 and 2. Each programmer will perform the specified command at almost the same instant. ConsoleEDS will wait until all specified programmers have completed the command before exiting and reporting PASS / FAIL.

The PASS / FAIL results of each programmer are displayed as follows:

Channel Result Error Condition

```
0  FAIL  Error 15 - Operation aborted
1  PASS  None
2  FAIL  Error 40 - SPI entering program mode failed
```

4.4.7 /AUTOPROGRAMBROADCAST command

This command is a special version of the /AUTOPROGRAM command which is used to start the execution of a specified '**Standalone programming project**' on all programmers on the network at exactly the same time. It is designed for use when programming panels of identical 'Target boards' and allows the simultaneous power-up and programming of each 'Target Board' on the panel.

PC command sent	
ConsoleEDS command	/AUTOPROGRAMBROADCAST
Command format	ConsoleEDS /AUTOPROGRAMBROADCAST=ProjectName /NODES=NumberOfNodes,Address0,AddressN
Command arguments	<p>where:</p> <ul style="list-style-type: none"> • ProjectName is the name of 'standalone programming project' to be executed on each programmer • NumberOfNodes is the number of programmers on the network. • Address0...AddressN is simply a list of the 'Node Addresses' of the programmers which the command is to be executed by.

Please note:

- This command broadcasts to all programmers on the network at the same time. It can therefore only be used when **all programmers** on the network are to be started at the same time.
- All programmers must contain the same projects.
- The specified '**standalone programming project**' must be in the same position (project number) in all programmers.

Example:

ConsoleEDS /AUTOPROGRAMBROADCAST=Project1 /NODES=4,0,1,2,3 /TIMEBEFOREPOLL=10000 /TOTALTIMEOUT=60

This command will execute the **"/AUTOPROGRAMBROADCAST=Project1"** command on 4 programmers at addresses 0, 1, 2 and 3. Each programmer will start execution of the project at exactly the same time. ConsoleEDS will wait the time specified by the command and will then start to poll all specified programmers until they have all finished. The command will then exit and report PASS or FAIL.

The PASS / FAIL results of each programmer are displayed as follows:

Channel Result Error Condition

```
0  FAIL  Error 15 - Operation aborted
1  PASS  None
2  FAIL  Error 40 - SPI entering program mode failed
3  PASS  None
```

4.4.8 /TIMEBEFOREPOLL command

The **/TIMEBEFOREPOLL** command is designed to be used with either the **/AUTOPROGRAM** or **/AUTOPROGRAMBROADCAST** commands. It specifies the time delay in milli-seconds between ConsoleEDS starting execution on all programmers to when it starts polling them to see whether each one has finished. The command can be used to minimise the communications overhead when programming panels of Target Boards.

4.4.9 /TOTALTIMEOUT command

This command is used to set the maximum amount of time in seconds that any ConsoleEDS command is allowed to take before automatically timing out. It is designed to be used when controlling multiple programmers from the same ConsoleEDS session and prevents ConsoleEDS hanging forever waiting for a programmer which may have got stuck to finish.

PC command sent	
ConsoleEDS command	/TOTALTIMEOUT
Command format	ConsoleEDS /OtherCommands /TOTALTIMEOUT=NumberOfSeconds
Command arguments	where: <ul style="list-style-type: none"> NumberOfSeconds is the number of seconds that ConsoleEDS will wait before timing out and reporting a "Timeout Error".
Example	ConsoleEDS BaseProject.prj /AUTOPROGRAM=Project1 /NODES=4,0,1,2,3 /TOTALTIMEOUT=60 This command will execute the "/AUTOPROGRAM=Project1" command on 3 programmers at addresses 0, 1 and 2. If this operation has not finished within e.g. 60 seconds (the "Timeout Period" then ConsoleEDS will exit with a "Timeout Error".

4.5 Standalone Programming Projects commands

4.5.1 Overview

This section describes the ConsoleEDS commands which are used to upload, execute and manage '**Standalone Programming Projects**'. These are projects which reside in the programmer '**FLASH Memory Store**' and which have been pre-compiled using EQTools - Project Builder or EDS.

ConsoleEDS supports uploading of a single project or a complete '**Project Collection**' to one or many attached programmers using the **/UPLOAD** command.

Example:

ConsoleEDS /UPLOAD=ProjectCollection.ppc

→ The Programming Projects contained in **ProjectCollection.ppc** to the attached programmer.

Once a project is resident in the programmer's memory, it is then possible to execute this project by specifying the '**Project UniqueID (name)**' with a single ConsoleEDS command.

Example:

ConsoleEDS /AUTOPROGRAM=PROJECT1

→ The Programming Project called '**Project1**' is executed.

This command mimics the action of executing the same project by pressing the **<Yes>** or **<START>** key on the programmer keypad.

There are also a number of '**Standalone Project**' housekeeping commands which allow a Remote Application to read a list of projects back from the programmer, check that a specified project is resident in the programmer etc. These commands are described in the next sections.

4.5.2 /UPLOAD command

This command allows a pre-compiled **Project Collection** file (*.ppc) to be uploaded to an attached programmer or multiple PPM3-MK2, PPM4-MK1 or ISPnano programmers on an RS485 network. This is a completely automated process. No human intervention is required during the upload process.

PC command sent									
ConsoleEDS command	/UPLOAD								
Command format	<i>ConsoleEDS /UPLOAD=ProjectCollection.ppc</i> The example will upload all Programming Projects contained in <i>ProjectCollection.ppc</i> to the attached programmer.								
Command arguments	<i>ProjectCollection.ppc (Project Collection File Name *.PPC)</i>								
ConsoleEDS debug output	<i>ConsoleEDS /UPLOAD=ProjectCollection.ppc</i> <i>Console EDS - Now Uploading and Verifying ProjectCollection.ppc...</i> <i>Console EDS - REBOOT_ME</i> <i>Operation: Project Upload and Verify</i> <i>Result: PASS</i> <i>Collection Name: ProjectCollection.PPC</i> <i>Number of projects: 4</i> <i>Path:</i> <i>Uploaded successfully to programmer(s):</i> <table><tr><th>Addr</th><th>Type</th><th>Firmware</th><th>Serial No</th></tr><tr><td>0</td><td>PPM3 MK2</td><td>3.07</td><td>763</td></tr></table> <i>Result: PASS</i> <i>Closing Console EDS, no errors have occurred.</i>	Addr	Type	Firmware	Serial No	0	PPM3 MK2	3.07	763
Addr	Type	Firmware	Serial No						
0	PPM3 MK2	3.07	763						

4.5.3 /PROJECTLIST command

This command returns a full textual list of all '**Standalone Programming Projects**' which are currently resident in the programmer '**FLASH Memory Store**'. The project '**Unique ID**' is returned which features the '**Project name**' plus version control characters e.g. "**PROJECT1_290601_1.1.1.1**".

PC command sent	
ConsoleEDS command	/PROJECTLIST
Command format	ConsoleEDS /PROJECTLIST
Command arguments	None
ConsoleEDS debug output	ConsoleEDS /PROJECTLIST project1_290601_1.1.1.1 project2_030601_1.1.1.3 project3_070601_1.2.1.9 Result: PASS Closing Console EDS, no errors have occurred.

4.5.4 /READPROJECTSTO command

This command reads the list of all "**Standalone Programming Projects**" which are currently resident in the programmer '**FLASH Memory Store**' and places them either in the Equinox '**Interface Database – Projects table**' or in a specified **TEXT File**. This allows ConsoleEDS to then use this cached Project List instead of reading the Project List every time a ConsoleEDS session is executed. This can save over 1 second per ConsoleEDS session if there are many projects stored in the programmer.

PC command sent	
ConsoleEDS command	/READPROJECTSTO
Command format	<p>Typical use of this command is as follows:</p> <p>/READPROJECTSTO=DB – reads the Project List to the 'Projects' table in the Interface Database.</p> <p>/READPROJECTSTO=TextFile.txt – reads the Project List to a text file called TextFile.txt</p>
Command arguments	Specify either ' DB ' for ' Database ' or a file name e.g. TextFile.txt for a text file
ConsoleEDS debug output	ConsoleEDS /READPROJECTSTO=ProjectList.txt Console EDS - Reading project list from programmer Console EDS - Finished reading project list from programmer Console EDS - Saving programmer projects to ProjectList.txt Result: PASS Closing Console EDS, no errors have occurred.

4.5.5 /GETPROJECTSFROM command

This command is used to specify to ConsoleEDS where it should read the list of '**Standalone Programming Projects**' from which are already resident in the programmer. If this command is not used then ConsoleEDS will read the '**Project List**' back for the programmer every time a Standalone Project is called via ConsoleEDS. This process can take more than a second to execute. If the /GETPROJECTSFROM command is used and the Project List is then read from a cached version either in the '**Interface Database**' or a Text File, then the process only takes typically 100 – 200ms.

PC command sent	
ConsoleEDS command	/GETPROJECTSFROM
Command format	<p>Typical use of this command is as follows:</p> <p>/GETPROJECTSFROM=DB – reads the Project List from the 'Projects' table in the Interface Database.</p> <p>/GETPROJECTSFROM=ProjectList.txt – reads the Project List from a text file.</p>
Command arguments	Specify either ' DB ' for ' Database ' or a file name e.g. TextFile.txt for a text file
ConsoleEDS debug output	None – This command is used in conjunction with other commands. e.g. /AUTOPROGRAM and /GOIMMEDIATEPROG

Please note:

The command is used in conjunction with any other command which uses a '**Standalone Programming Project**' as an argument.

e.g. **/AUTOPROGRAM**, **/GOIMMEDIATEPROG**, **/CHECKPROJECT**

Examples:

ConsoleEDS /CHECKPROJECT=PROJECT1 /GetProjectsFrom=ProjectList.txt

ConsoleEDS UTRF-169V.prj /AUTOPROGRAM=PROJECT1

/GETPROJECTSFROM=ProjectList.txt

Please note:

The use of the **/GetProjectsFrom** command can significantly speed up repetitive ConsoleEDS command scripts as the '**Project List**' is now stored locally in the text file so it does not have to be downloaded from the programmer each time.

4.5.6 /CHECKPROJECT command

This command is used to check whether a specified '**Standalone Programming Project**' is resident in the Programmer '**FLASH Memory Store**'.

PC command sent	
ConsoleEDS command	/CHECKPROJECT
Command format	ConsoleEDS /CHECKPROJECT=PROJECT1 This command would check whether the project ' PROJECT1 ' is resident in the programmer.
Command arguments	ProjectName This is the name of the ' Standalone Project ' without any file extension e.g. ' PROJECT1 '
ConsoleEDS debug output	ConsoleEDS /CHECKPROJECT=PROJECT1 Console EDS - Checking Project 'PROJECT1' is in the Programmer FLASH Memory Store Console EDS - Loading programmer projects from ProjectList.txt Console EDS - Project 'PROJECT1' found in programmer at address 2

Please note:

This command is usually used with the **/GetProjectsFrom=ProjectList.txt** command.

e.g. **ConsoleEDS /CHECKPROJECT=PROJECT1 /GetProjectsFrom=ProjectList.txt**

4.5.7 /AUTOPROGRAM command

This command initiates the execution of the specified '**Standalone Programming Project**'. The ProjectName is the full name (Unique ID) of a Programming Project which is resident in the programmer.

PC command sent	
ConsoleEDS command	/AUTOPROGRAM
Command format	/AUTOPROGRAM=PROJECTNAME
Command arguments	ProjectName e.g. PROJECT1
ConsoleEDS debug output (typical example)	<p>ConsoleEDS /AUTOPROGRAM=PROJECT1</p> <p>Console EDS - Autoprogram project : PROJECT1</p> <p>Console EDS - Loading programmer projects from ProjectList.txt</p> <p>Preparing to power up target</p> <p>Applying power to target</p> <p>Waiting for internal erase</p> <p>Programming flash memory</p> <p>Programmer is Waiting</p> <p>Result: PASS</p> <p>Closing Console EDS, no errors have occurred.</p>

Please note:

- The Project Name must be specified as the full '**Unique Project ID**' including all of the version control characters. e.g. **PROJECT1_290601_1.2.3.4**. No file extension should be used.
- This command is usually used with the **/GetProjectsFrom=ProjectList.txt** command. e.g. **ConsoleEDS /AUTOPROGRAM=PROJECT1 /GetProjectsFrom=ProjectList.txt**
- The **/AUTOPROGRAM** command can be used in conjunction with other programming commands within the same ConsoleEDS session.
- When setting up a Standalone Project for ConsoleEDS, it is necessary to select the "**Project Type**" as "**ISP-PRO – End in programming mode**". This setting means that once the Standalone Project has executed it will keep the Target System powered up and in "programming mode" so that the next ConsoleEDS command can then be executed. This can save a lot of time in production as powering down and then up again and re-entering programming mode can take up to a second.

Related commands:

The **/AUTOPROGRAMBROADCAST** command operates in the same manner as the **/AUTOPROGRAM** command except that it starts all programmers on a network at exactly the same time.

4.5.8 /GOIMMEDIATEPROG command

This command initiates the execution of the specified '**Standalone Programming Project**'. It differs from the **/AUTOPROGRAM** command in that it does NOT power up, enter programming mode etc. It assumes that the Target System is already powered up and that the Target IC is already in 'Programming Mode' and so simply carries on with the first programming action of the '**Standalone Programming Project**'. This command is used to stop the programmer exiting 'Programming Mode', powering down, and powering back up again during complicated ConsoleEDS programming sequences. This will help to reduce the overall programming time of the sequence and power-cycling can take e.g. 1 second due to the requirement to discharge / charge the power supply capacitance on many Target Boards.

PC command sent	
ConsoleEDS command	/GOIMMEDIATEPROG
Command format	/GOIMMEDIATEPROG=PROJECTNAME
Command arguments	PROJECTNAME e.g. PROJECT1
ConsoleEDS debug output (typical example)	<p>ConsoleEDS /GOIMMEDIATEPROG=PROJECT1</p> <p>Console EDS - Autoprogram project : PROJECT1</p> <p>Console EDS - Loading programmer projects from ProjectList.txt</p> <p>Waiting for internal erase</p> <p>Programming flash memory</p> <p>Programmer is Waiting</p> <p>Result: PASS</p> <p>Closing Console EDS, no errors have occurred.</p>

Please note:

- The Project Name must be specified as the full '**Unique Project ID**' including all of the version control characters. e.g. **PROJECT1_290601_1.2.3.4**. No file extension should be used.
- This command is usually used with the **/GetProjectsFrom=ProjectList.txt** command. e.g. **ConsoleEDS /GOIMMEDIATEPROG=PROJECT1 /GetProjectsFrom=ProjectList.txt**
- The **/GOIMMEDIATEPROG** command can be used in conjunction with other programming commands within the same ConsoleEDS session.
- When setting up a Standalone Project for ConsoleEDS, it is necessary to select the "**Project Type**" as "**ISP-PRO – End in programming mode**". This setting means that once the Standalone Project has executed it will keep the Target System powered up and in "programming mode" so that the next ConsoleEDS command can then be executed. This can save a lot of time in production as powering down and then up again and re-entering programming mode can take up to a second.

4.6 Power control + Enter / Exit Programming Mode commands

4.6.1 Overview

This section describes the ConsoleEDS commands used to reset the programmer / Target System and to enter and exit 'Programming Mode'

4.6.2 /POWERUP command

This command instructs the programmer to "power up" the Target System (UUT) without entering programming mode. It can be used to power the UUT up to a specified voltage for eg. a functional test where the DUT is not to be programmed. The same command can also be used to control the '**EXTERNAL VCC SWITCH**' on the ISPnano programmer and can therefore also control the relays on the '**ISPnano Series 3 ATE**' programmer.

PC command sent	
ConsoleEDS command	/POWERUP
Command format	ConsoleEDS BaseProject.prj /POWERUP=2,3,4 /NORESET → The programmer will power up the UUT using the power supply settings specified in the Base Project. → The programmer will also automatically measure and display the ' Target Vcc ', ' Target Current ', ' External Vcc – Input ' and, ' External Vcc – Output ' depending on the arguments specified.
Command arguments	The numbers after the command specify which voltages are to be measured: 2 - 'Target Vcc' (default) 3 - 'External Vcc – Input' 4 - 'External Vcc – Output' The /NORESET command is used to stop the programmer powering down the UUT at the end of the ConsoleEDS session.
Typical debug output	Console EDS - Target Voltage: 3.30 V Console EDS - External Vcc Input Voltage: 11.81 V Console EDS - External Vcc Output Voltage: 11.81 V Console EDS - Target Current: 1.6 mA

Please note:

- To switch power off to the UUT, the **/RESET** command should be used.
- This command will only work when the programmer is controlling 'Target Power' or controlling the '**EXTERNAL VCC SWITCH**' on an ISPnano programmer.
- The measured current is from the '**Programmer controlled power supply**' to the UUT.
- This command was introduced in ConsoleEDS build 2131.

4.6.3 /RESET command

This command will reset the attached programmer.

It will force all the programmer I/O pins to tri-state and if the programmer is controlling power to the Target System, then the '**Programmer Controlled Power supply**' will be switched off. It will also switch off the '**EXT-VCC Switch**' on the '**ISPnano Series 3 ATE**' programmer.

PC command sent	
ConsoleEDS command	/RESET
Command format	→ The programmer will tri-state all the programmer I/O pins → If the programmer was powering the Target System (PPM3-MK2 only) then the Target System will be powered off
Command arguments	None

Please note:

By default, ConsoleEDS will automatically perform a **/RESET** at the end of each ConsoleEDS session. This is a safety mechanism to make sure the Target System is not left powered up and in programming mode at the end of a sequence. For some more complex programming sequences it may be desirable to keep the Target Device in programming mode. The **/NORESET** command is used to keep the Target Device in programming mode at the end of a session.

4.6.4 /NORESET command

This command is used to force the programmer to keep the Target System powered up and to stay in 'Programming Mode' at the end of a ConsoleEDS session. This allows the next ConsoleEDS session to start with the Target IC already powered up and in programming mode. In complex programming sequences this can save a lot of time.

PC command sent	
ConsoleEDS command	/NORESET
Command format	ConsoleEDS BaseProject.prj /COMMAND1 /COMMAND2 /NORESET → The programmer will execute "COMMAND1" followed by "COMMAND2" and then keep the Target System powered up and the Target IC will remain in programming mode.
Command arguments	None
ConsoleEDS debug output	None

4.6.5 /FASTSTART command

This command tells ConsoleEDS that the Target IC is already in 'Programming Mode'. The actions specified on the ConsoleEDS command line will then be carried out without first trying to re-enter programming mode. This can help to reduce the overall programming time of a complicated

Programming Sequence as the Target IC does not have to exit and re-enter Programming Mode every time a new ConsoleEDS command is executed.

PC command sent	
ConsoleEDS command	/FASTSTART
Command format	/FASTSTART
Command arguments	None
ConsoleEDS debug output	<p>Example:</p> <p>ConsoleEDS BaseProject.prj /FASTSTART /EEPROMWRITE=eefile.hex</p> <p>This command sequence will assume the Target IC is already in Programming Mode and will immediately execute the next command on the Command Line. e.g. /EEPROMWRITE.</p>

Please note:

- The previous ConsoleEDS session (executed before this session) must have left the Target IC powered up and in Programming Mode otherwise the **/FASTSTART** command will fail.

4.6.6 /AUTO_INIT command

The new command **/AUTO_INIT** has been introduced in ConsoleEDS build 1119. This command gives the option of using a new firmware command to speed up the entry into programming mode with many algorithms. This command performs the target power-up, pre-programming statemachine and entry into programming mode operations under programmer control instead of PC control. This makes the process of entering programming mode significantly faster for most algorithms as the PC is no longer wasting time polling the programmer.

The **/AUTO_INIT** can be used with any ConsoleEDS command which would need to enter programming mode.

Example:

ConsoleEDS /ERASE /FLASHWrite=FLASHFile.bin /AUTO_INIT

This command session would use the new faster method to enter programming mode and it would then erase the FLASH and then program in the specified file.

Please note:

- This feature is currently only available on the ISPnano programmer with firmware 5.14 or above.

4.7 Checking the Device Signature and JTAG ID

4.7.1 Overview

This section describes how to check the '**Device Signature**' of an SPI / JTAG device and / or a '**JTAG ID**' of a JTAG device. Most SPI programmable devices feature a so called '**Device Signature**' which allows the programmer to check that the Target IC is the correct device before programming it, All JTAG programmable devices feature a so called 4-byte '**JTAG ID**' which not only identifies the Target IC but also gives the '**Silicon Revision**' of the device.

Some Atmel ATmega AVR microcontrollers feature both a '**Device Signature**' and a '**JTAG ID**' in JTAG programming mode. The **/READSIG** command can be used to check either the '**Device Signature**' or '**JTAG ID**'. This can be set up in the Base Project by specifying 'Read Device Signature' and/or 'Read JTAG ID' (See 'Device' tab in Project Builder or EDS).

Please note:

ConsoleEDS will automatically check the **Device Signature / JTAG ID** of the Target IC when initially entering programming mode. It is usually not necessary therefore to use the **/READSIG** command.

4.7.2 /READSIG command (Device Signature)

This command returns the Device Signature (Device ID) of any Target IC which features an on-chip electronic '**Device Signature / ID**'. The format of returned Device Signature differs between device types.

PC command sent	
ConsoleEDS command	/READSIG
Command format	ConsoleEDS BaseProject.prj /READSIG Where <ul style="list-style-type: none"> • BaseProject.prj is the Base Project for the Target IC.
Command arguments	None
ConsoleEDS debug output	ConsoleEDS will display a different 'Device ID' format depending on the Target Device. Typical response: Signature read back: 0x1E9007

4.7.3 /READSIG command (Device JTAG ID)

This command returns the '**JTAG ID**' of the specified Target JTAG IC. The format of returned ID is the generic 4-byte JTAG ID. This command will work for any JTAG IC as long as the relevant JTAG parameters are set up in the Base Project.

PC command sent	
ConsoleEDS command	/READSIG
Command format	ConsoleEDS ATmega128.prj /READSIG Where <ul style="list-style-type: none"> • ATmega128.prj is the Base Project for the Target IC.
Command arguments	None
ConsoleEDS debug output	ConsoleEDS will display the 4-byte 'JTAG ID' of the Target IC. Typical response: Console EDS - Checking signature is 0x1E9405 JTAG ID: 0x4940503F Revision: 4 Part Number: 0x9405 Manufacturer: 0x001F

4.8 /ERASE command

This command performs a '**Chip Erase**' of the Target IC.

The exact outcome of this command depends on the actual IC being programmed.

For most devices, this command will erase the FLASH + EEPROM contents to 0xFF value and then erase the 'Security Fuses' so the device can then be re-programmed.

Please note:

When using ConsoleEDS Standard Version, a Chip Erase is always carried out.

The '**Chip Erase**' option must be **ENABLED** in the 'Base Project' otherwise the Erase Operation will not be carried out.

PC command sent	
ConsoleEDS command	/ERASE
Command format	<i>ConsoleEDS /ERASE</i>
Command arguments	None
ConsoleEDS debug output	<i>ConsoleEDS /ERASE</i> <i>00:01.375 Console EDS - Erasing target</i> <i>00:01.375 Console EDS - TARGET_ERASE</i> <i>Result: PASS</i> <i>Closing Console EDS, no errors have occurred.</i>

4.9 FLASH Area – WRITE, READ and VERIFY commands

4.9.1 Overview

This section details the commands required to write (program) data into, read data from and to verify data against the FLASH area of a Target IC. The table below details the commands discussed in this section.

ConsoleEDS command	Description
/FLASHWRITE	Programs the contents of a Binary or Hex file into the FLASH area of a Target IC.
/FLASHVERIFY	Verifies the contents of a Binary or Hex file against the FLASH area of a Target IC.
/FLASHREAD	Reads the data from a specified range of the FLASH area of a Target IC to a Binary or Hex file.

The '**command arguments**' are the same for all of the commands in this section and are detailed in the table below.

Command arguments	
	<ul style="list-style-type: none"> • Baseproject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc. • FileName.bin is the file name of a BINARY data file • FileName.hex is the file name of an Intel Hex data file • From is the START Address in the Target IC to start programming / verifying / reading from. • To is the END Address in the Target IC for the data to end programming / verifying / reading at. • Offset is the offset number of bytes into the data file.

4.9.2 /FLASHWRITE (entire file) command

This command programs the entire contents of the specified file into the '**FLASH**' (**CODE**) area of the Target IC. It also performs a verify operation on a page-by-page basis as it is programming.

PC command sent	
ConsoleEDS command	/FLASHWRITE=FileName (Writes entire file)
Command format	/FLASHWRITE=FileName (Writes entire file) Example: ConsoleEDS baseproject.prj /FLASHWRITE=FileName.bin or ConsoleEDS baseproject.prj /FLASHWRITE=FileName.hex

4.9.3 /FLASHWRITE (specified address range) command

This command programs the data from a file into a specified address range in the '**FLASH**' (**CODE**) area of the Target IC. It is possible to also specify an **offset** into the data file so ConsoleEDS will start reading the data from an address in the file other than zero. This command also performs a verify operation on a page-by-page basis as it is programming.

PC command sent	
ConsoleEDS command	/FLASHWRITE=FileName,From,To,FileOffset
Command format	<p>Example: ConsoleEDS baseproject.prj /FLASHWRITE=serial5.bin,10,15,0</p> <p>This example would program/verify the contents of the specified file 'serial5.bin' into the FLASH area of the Target IC from address 10d to 15d.</p>

4.9.4 /FLASHVERIFY (entire file) command

This command verifies the entire contents of the specified file against the '**FLASH**' (**CODE**) area of the Target IC.

PC command sent	
ConsoleEDS command	/FLASHVERIFY=FileName (Verifies entire file)
Command format	<p>/FLASHVERIFY =FileName (Verifies entire file)</p> <p>Example: /FLASHVERIFY baseproject.prj /FLASHVERIFY=FileName.bin or ConsoleEDS baseproject.prj /FLASHVERIFY=FileName.hex</p>
Command arguments	FileName (Verifies entire file)

4.9.5 /FLASHVERIFY (specified address range) command

This command verifies the data from a file against a specified address range in the '**FLASH**' (**CODE**) area of the Target IC. It is possible to also specify an offset into the data file so ConsoleEDS will start verifying the data from address other than zero.

PC command sent	
ConsoleEDS command	/FLASHVERIFY=FileName,From,To,FileOffset
Command format	<p>Example: ConsoleEDS baseproject.prj /VERIFY=serial5.bin,10,15,0</p> <p>This example would verify the contents of the specified file 'serial5.bin' against the FLASH area of the Target IC from address 10 to 15.</p>

4.9.6 /FLASHREAD (specified address range) command

This command reads a block of data from the FLASH area of a Target IC and saves it to a file.

PC command sent	
ConsoleEDS command	/FLASHREAD=FileName,From,To
Command format	/FLASHREAD=FileName,From,To Example: ConsoleEDS baseproject.prj /FLASHREAD=ReadOutData.bin,10,18 This example would read the data from FLASH address 10d to 18d and place it in a file called ReadOutData.bin .

4.9.7 /RPLB (AT91SAM7 - Read FLASH Page Lock Bits) command

A new command **/RPLB** has been introduced in ConsoleEDS build 1119. This command reads back the '**FLASH Page Lock Bits**' from any AT91SAM7 device and returns them as 4 hexadecimal bytes eg. **0x00 0x00 0x00 0x00**.

PC command sent	
ConsoleEDS command	/RPLB
Command format	BaseProject.prj /RPLB
Command arguments	<ul style="list-style-type: none"> Baseproject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc.
ConsoleEDS debug output	0x00 0x00 0x00 0x00

Please note:

- This command will only work with Atmel AT91SAM7 devices.
- This feature is currently only available on the ISPnano programmer with firmware 5.14 or above.

4.10 EEPROM Area – WRITE, READ and VERIFY commands

4.10.1. Overview

This section details the commands required to write (program) data into, read data from and to verify data against the EEPROM area of a Target IC. The table below details the commands discussed in this section.

ConsoleEDS command	Description
/EEPROMWRITE	Programs the contents of a Binary or Hex file into the EEPROM area of a Target IC.
/EEPROMVERIFY	Verifies the contents of a Binary or Hex file against the EEPROM area of a Target IC.
/EEPROMREAD	Reads the data from a specified range of the EEPROM area of a Target IC to a Binary or Hex file.

The '**command arguments**' are the same for all of the commands in this section and are detailed in the table below.

Command arguments	
	<ul style="list-style-type: none"> • Baseproject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc. • FileName.bin is the file name of a BINARY data file • FileName.hex is the file name of an Intel Hex data file • From is the START Address in the Target IC to start programming / verifying / reading from. • To is the END Address in the Target IC for the data to end programming / verifying / reading at. • Offset is the offset number of bytes into the data file.

4.10.2 /EEPROMWRITE (entire file) command

This command programs the entire contents of the specified file into the '**EEPROM**' (**DATA**) area of the Target IC. It also performs a verify operation on a byte-by-byte or page-by-page basis as it is programming.

PC command sent	
ConsoleEDS command	/EEPROMWRITE=FileName (Writes entire file)
Command format	/EEPROMWRITE=FileName (Writes entire file) Example: ConsoleEDS baseproject.prj /EEPROMWRITE=FileName.bin or ConsoleEDS baseproject.prj /EEPROMWRITE=FileName.hex

4.10.3 /EEPROMWRITE (specified address range) command

This command programs the data from a file into a specified address range in the '**EEPROM**' (**DATA**) area of the Target IC. It is possible to also specify an offset into the data file so ConsoleEDS will start reading the data from address other than zero. This command also performs a verify operation on a page-by-page basis as it is programming.

PC command sent	
ConsoleEDS command	/EEPROMWRITE=FileName,From,To,FileOffset
Command format	/EEPROMWRITE=FileName,From,To,FileOffset Example: ConsoleEDS baseproject.prj /EEPROMWRITE=serial5.bin,10,15,0 This example would program/verify the contents of the specified file ' serial5.bin ' into the EEPROM area of the Target IC from address 10 to 15.

4.10.4 /EEPROMVERIFY (entire file) command

This command programs the entire contents of the specified file into the '**EEPROM**' (**DATA**) area of the Target IC. It also performs a verify operation on a byte-by-byte or page-by-page basis as it is programming.

PC command sent	
ConsoleEDS command	/EEPROMVERIFY=FileName (Writes entire file)
Command format	/EEPROMVERIFY =FileName (Writes entire file) Example: ConsoleEDS baseproject.prj /EEPROMVERIFY =FileName.bin or ConsoleEDS baseproject.prj /EEPROMVERIFY=FileName.hex

4.10.5 /EEPROMVERIFY (specified address range) command

This command verifies the data from a file against a specified address range in the '**EEPROM**' (**DATA**) area of the Target IC. It is possible to also specify an offset into the data file so ConsoleEDS will start verifying the data from address other than zero.

PC command sent	
ConsoleEDS command	/EEPROMVERIFY=FileName,From,To,FileOffset
Command format	/EEPROMVERIFY=FileName,From,To,FileOffset Example: ConsoleEDS baseproject.prj /EEPROMVERIFY=serial5.bin,10,15,0 This example would verify the contents of the specified file ' serial5.bin ' against the EEPROM area of the Target IC from address 10 to 15.

4.10.6 /EEPROMREAD (specified address range) command

This command reads a block of data from the EEPROM area of a Target IC and saves it to a file.

PC command sent	
ConsoleEDS command	/FLASHREAD=FileName,From,To
Command format	/FLASHREAD=FileName,From,To Example: ConsoleEDS baseproject.prj /EEPROMREAD=ReadOutData.bin,10,18 This example would read the data from EEPROM address 10d to 18d and place it in a file called ReadOutData.bin .

4.11 Programming 'Configuration Fuses' commands

4.11.1 Overview

This section describes how to program the '**Configuration Fuses**' of a Target IC.

The commands which support programming / reading the fuses are detailed in the table below.

ConsoleEDS command	Description
/WRITE...FUSES From a Fuse File	These commands use the specified 'Pre-Erase', 'Post-Erase' or 'Pre-lock' fuse write from the project to program the fuses of the Target IC. The fuse values must be pre-compiled into a 'Fuse File' (*.eff).
/WRITE...FUSES using HEX values	Same as above except the command supports programming of the fuses of the Target IC by simply specifying the Fuse Values in HEX.
/FUSEWRITE	This command supports programming of the fuses of the Target IC by simply specifying the Fuse Values in HEX without needing to specify which 'fuse write' to use.
/READFUSES	This command reads the fuses of a Target IC and displays them in HEX format.

4.11.2 /WRITE.....FUSES command using a 'Fuse File'

There are 3 different Fuse Write commands in a 'Standalone Programming Project' which allow the Fuses of a Target IC to be programmed either before the Chip Erase, after the Chip Erase or before the Security Fuses are programmed. The choice of Fuse Write operation depends on the Chip Family and also the desired effect of the Fuse Write.

PC command sent	
ConsoleEDS command	There are three variations of this command: /WRITE_PRE_ERASE_FUSES=FuseFile.prj /WRITE_POST_ERASE_FUSES=FuseFile.prj WRITE_PRE_LOCK_FUSES= FuseFile.prj
Command format	ConoleEDS BaseProject.prj /WRITE_POST_ERASE_FUSES=FuseFile.prj This command performs a Program/Verify of the Configuration Fuses of the Target IC. These Fuse settings are taken from the <Post-erase Fuse> settings of a compiled Programming Project (*.prj) Example: ConsoleEDS baseproject.prj /WRITE_POST_ERASE FUSES=FuseFile.prj Where: <ul style="list-style-type: none"> Baseproject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc.

	<ul style="list-style-type: none"> FuseFile.prj is a compiled Programming Project (*.prj) which contains the required Fuse Settings. <p>It is possible to use the same project for both the 'Base Project' and 'Fuse Project'.</p>
Command arguments	FuseFile.prj

4.11.3 /WRITE.....FUSES command using 'HEX Fuse Values'

It is possible to program the fuses of a Target IC by simply specifying the '**Fuse HEX values**' on the command line. This method allows programming of the fuses without having to use a '**Fuse File**'. If a project for an AVR microcontroller has been developed using 'AVR Studio' then the '**Fuse Hex values**' from this software can be used in ConsoleEDS.

PC command sent	
ConsoleEDS command	/WRITE.....FUSES
Command format	ConsoleEDS BaseProject.prj /POSTERASEFUSEWRITE=0x42,0xA9,0xFF This command performs a Program/Verify of the Configuration Fuses using the 'Fuse Hex values' specified on the command line.
Command arguments	FuseHexValue1, FuseHexValue2...FuseHexValuen
ConsoleEDS debug output	ConsoleEDS BaseProject.prj /POSTERASEFUSEWRITE=0x42,0xA9,0xFF Console EDS - Fuse Bytes - Program / Verify Console EDS - MISP_FUSE_WRITE Console EDS - MISP_FUSE_READ Console EDS - Read Fuses 0x42 0xA9 0xFF 0x00 0x00

Please note:

The relevant '**Fuse Write**' must be enabled in the **Base Project**.

So, if the **<Post Erase Fuse Write>** is to be used, then this operation must be enabled in the **Base Project**.

4.11.4 /FUSEWRITE command using 'HEX Fuse Values' (build 2008)

A new **/FUSEWRITE** command was introduced in build 2008 onwards which makes it possible to program the fuses of a Target IC by simply specifying the '**Fuse HEX values**'. There is no requirement to specify which 'Fuse Write' to use...ConsoleEDS does this automatically.

PC command sent	
ConsoleEDS command	/FUSEWRITE
Command format	ConsoleEDS BaseProject.prj /FUSEWRITE=0x42,0xA9,0xFF This command performs a Program/Verify of the Configuration Fuses using the 'Fuse Hex values' specified on the command line.

Command arguments	FuseHexValue1, FuseHexValue2...FuseHexValuen
ConsoleEDS debug output	<p><i>ConsoleEDS BaseProject.prj /FUSEWRITE=0x42,0xA9,0xFF</i></p> <p>Console EDS - Fuse Bytes - Program / Verify Console EDS - MISP_FUSE_WRITE Console EDS - MISP_FUSE_READ Console EDS - Read Fuses 0x42 0xA9 0xFF 0x00 0x00</p>

4.11.5 /READFUSES command

This command reads the values of the 'Configuration Fuses' back from the Target IC. The resulting data can then either be displayed on-screen as hexadecimal values or written to a Binary file.

PC command sent	
ConsoleEDS command	/READFUSES
Command format	<p>i. /READFUSES This command will read the value of the 'Configuration Fuses' of the Target IC and display them on-screen.</p> <p>Example: <i>ConsoleEDS baseproject.prj /READFUSES</i></p> <p>A typical response would be: <i>Console EDS - Read Fuses 0x6A 0xFF 0x00 0x00 0x00 0x00</i></p> <p>ii. /READFUSES=FuseFile.bin This command will read the value of the 'Configuration Fuses' of the Target IC and write them to a binary file.</p> <p>Example: <i>ConsoleEDS baseproject.prj /READFUSES=FuseFile.bin</i></p>
Command arguments	<ul style="list-style-type: none"> • Baseproject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc. • FuseFile.bin is a binary file where the fuse values will be written to. The Fuses are written in Binary format so the file is 6 bytes long if there are 6 x Fuse Bytes. The file will be automatically created if it does not already exist.

4.12 Security Fuse (Lock Bits) programming commands

4.12.1 Overview

This section describes how to program the 'Security Fuses' of a Target IC.

4.12.2 /WRITESECURITY command using a Fuse File

This command performs a Program/Verify of the Security Fuses (Lock Bits) of the Target IC. These Fuse settings are taken from the **<Security>** settings of a compiled Programming Project (*.prj).

PC command sent	
ConsoleEDS command	/WRITESECURITY
Command format	ConsoleEDS BaseProject.prj /WRITESECURITY=SecurityFile.prj
Command arguments	<ul style="list-style-type: none"> • Baseproject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc. • FuseFile.prj is a compiled Programming Project (*.prj) which contains the required Fuse Settings. <p>It is possible to use the same project for both the 'Base Project' and 'Security Fuse Project'</p>
ConsoleEDS debug output	ConsoleEDS BaseProject.prj /WRITESECURITY=SecurityFile.prj Console EDS - Writing security fuses Console EDS - Reading security fuses Console EDS - Security Fuses 0xFC Console EDS - Security Verify PASS Result: PASS Closing Console EDS, no errors have occurred.

4.12.3 /WRITESECURITY command using 'Hex Fuse Bytes'

This command performs a Program/Verify of the Security Fuses (Lock Bits) of the Target IC using the 'Hex Fuse Bytes' specified on the command line.

PC command sent	
ConsoleEDS command	/WRITESECURITY=FuseValue1, FuseValue2 etc
Command format	ConsoleEDS BaseProject.prj /WRITESECURITY=FuseHexValue
Command arguments	Baseproject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc.

	FuseHexValue This is the Hex value to be written to the Security Fuse(s) e.g. 0xFC
ConsoleEDS debug output	ConsoleEDS BaseProject.prj /WRITESECURITY=0xFC Console EDS - Writing security fuses Console EDS - Reading security fuses Console EDS - Security Fuses 0xFC Console EDS - Security Verify PASS Result: PASS Closing Console EDS, no errors have occurred.

4.12.4 /READSECURITY command

This command reads the values of the 'Security Fuses' back from the Target IC. The resulting data can then either be displayed on-screen as hexadecimal values or written to a Binary file.

PC command sent	
ConsoleEDS command	/READSECURITY
Command format	<p>i. /READSECURITY This command will read the value of the 'Configuration Fuses' of the Target IC and display them on-screen.</p> <p>Example: ConsoleEDS baseproject.prj /READSECURITY</p> <p>Where:</p> <ul style="list-style-type: none"> Baseproject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc. <p>A typical response would be: Console EDS - Security Fuses 0xFF 0x00 0x00 0x00</p> <p>ii. /READSECURITY=FuseFile.bin This command will read the value of the 'Configuration Fuses' of the Target IC and write them to a binary file.</p> <p>Example: ConsoleEDS baseproject.prj /READSECURITY=SecurityFile.bin</p> <p>Where:</p> <ul style="list-style-type: none"> Baseproject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc. SecurityFile.bin is a binary file where the Security Byte values will be written to. The Security Bytes are written in Binary format. The file will be automatically created if it does not already exist.
Command arguments	<ul style="list-style-type: none"> Baseproject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc. SecurityFile.bin is a binary file where the Security Byte values will be written to. The Security Bytes are written in Binary format. The file will be automatically created if it does not already exist.

4.13 /READCAL command (AVR microcontrollers only)

This command is for use with Atmel AVR Microcontrollers only. It reads the factory programmed '**Calibration Byte**' of the **Internal Oscillator** and then writes this byte into a location in either the FLASH or EEPROM.

PC command sent	
ConsoleEDS command	/READCAL=Address, FLASH or EEPROM
Command format	<i>ConsoleEDS BaseProject.prj /READCAL=Address, EEPROM</i>
Command arguments	<ul style="list-style-type: none"> • Baseproject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc. • Address is the physical address in the FLASH or EEPROM where the Calibration Byte is to be written. • EEPROM specifies that the Calibration Byte is to be written into the EEPROM area.

4.14 Zensys ZWxxx – Read / Write HomeID command

4.14.1 Overview

This section describes commands which are used to read / write the '**HomeID**' parameter of Zensys ZWxxxx microcontrollers.

4.14.2 /SETHOMEID command

This command is for use with Zensys ZW0101 and ZW0201 microcontrollers. It allows the 'Zensys HomeID' parameter to be written to the Target IC. This is a 4-byte number which can be used as a unique identifier or serial number for the IC.

PC command sent	
ConsoleEDS command	/SETHOMEID
Command format	<i>ConsoleEDS BaseProject.prj /SETHOMEID=SerialNumber</i>
Command arguments	<i>HomeID Number (4-byte long integer)</i>
ConsoleEDS debug output	

4.14.3 /READHOMEID command

This command is for use with Zensys ZW0101 and ZW0201 microcontrollers. It allows the 'Zensys HomeID' parameter to be read from a Target IC. This is a 4-byte number which can be used as a unique identifier or serial number for the IC.

PC command sent	
ConsoleEDS command	/READHOMEID
Command format	<i>ConsoleEDS BaseProject.prj /READHOMEID</i>
Command arguments	<i>BaseProject.prj</i>
ConsoleEDS debug output	

4.15 Scripting commands

4.15.1 Overview

This section describes the commands used to execute '**Programming Scripts**' which have been generated by the **EQTools – Script Builder** utility. Use of the **SCRIPT** and **INSERT** commands requires a chargeable upgrade for the 'Script Builder' utility.

4.15.2 /SCRIPT command

This is special command which allows a pre-compiled 'Programming Script' to be executed from the Command Line. This /SCRIPT command supports complex programming sequences which would be otherwise impossible if just using Command Line arguments. It is also possible to log each programming iteration to the Interface Database using a Programming Script.

PC command sent	
ConsoleEDS command	/SCRIPT=ScriptName.esf
Command format	ConsoleEDS BaseProject.prj /SCRIPT=ScriptName.esf
Command arguments	<ul style="list-style-type: none"> BaseProject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc. ScriptName.esf is a compiled Programming Project (*.prj)
ConsoleEDS debug output	Output depends on the script being executed

4.15.3 /INSERT command

This is special command which allows a single line of 'Programming Script' to be executed from the Command Line. It can be used to perform a complex programming action such as programming a Serial Number which would be otherwise impossible with the standard ConsoleEDS CommandSet.

PC command sent	
ConsoleEDS command	/INSERT="Line of script"
Command format	Example: ConsoleEDS BaseProject.prj /INSERT="MISPEepromWrite 0x0000 2 WORD #INC0"
Command arguments	<ul style="list-style-type: none"> BaseProject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc. MISPEepromWrite instructs the programmer to write to the EEPROM area 0x0000 is the start address to write from '2 WORD' is the number of bytes in a WORD #INC0 is the Incremental Number
ConsoleEDS debug output	Output depends on the script being executed.

4.15.4 /NOWAITKEY command

This is special argument for use with any Programming Script which waits for a user to press the <Connect> / <Disconnect> button in order to start / stop a programming cycle. If the argument is specified, then when the script is executed in ConsoleEDS, the PC will not wait for a key to be pressed. This allows the same script to be used in both ISP-PRO and ConsoleEDS.

PC command sent	
ConsoleEDS command	/SCRIPT=ScriptName.esf /NOWAITKEY
Command format	ConsoleEDS /SCRIPT=ScriptName.esf /NOWAITKEY
Command arguments	ScriptName.esf
ConsoleEDS debug output	

4.16 /SETINC command

This is special command which allows the settings for writing an Incremental Number or MAC address to be initialised.

PC command sent	
ConsoleEDS command	/SETINC=Incremental Source Number, CurrentValue, Increment, MaximumValue, Format
Command format	ConsoleEDS /SETINC=1,1000,1,2000,%d
Command arguments	<p>Where:</p> <ul style="list-style-type: none"> • 'Incremental Source Number' is the name of the incremental number as declared in EQTools – Incremental Repository. • 'Current Value' is the start value of the eg. serial number • 'Increment' is the increment to be added to the eg. serial number • 'Maximum Value' is the maximum value of the eg. serial number. • 'Format' is the data format of the incremental number. <p>The possible data Formats are: %d = Decimal, %X = Hex.</p> <p>This example would set up the 'Incremental Repository' to generate a number starting at 1000 decimal which would be incremented by 1 each time the relevant Write Command was executed.</p>
ConsoleEDS debug output	

4.17 /TIME command

This is special command which can be used during the debugging of a ConsoleEDS application. If the **/TIME** parameter is used, ConsoleEDS will then output a '**Time Stamp**' next to each command as it is executed.

PC command sent	
ConsoleEDS command	/TIME
Command format	<i>ConsoleEDS BaseProject.prj /Command1 / Command2 /TIME</i>
Command arguments	None
ConsoleEDS debug output	<i>00:01.375 Console EDS - Erasing target</i> <i>00:01.375 Console EDS - TARGET_ERASE</i> <i>00:01.390 Console EDS - SYSTEM_STATUS</i> <i>00:01.406 Console EDS - MISP_PRE_PROGRAMING_MODE</i> <i>00:01.437 Console EDS - MISP_ISP_INIT</i> <i>00:01.453 Console EDS - Fuse Bytes - Program / Verify</i> <i>00:01.469 Console EDS - MISP_FUSE_WRITE</i> <i>00:01.515 Console EDS - MISP_FUSE_READ</i> <i>00:01.547 Console EDS - Read Fuses 0x42 0xA9 0xFF 0x00 0x00</i>

4.18 /MEASUREVOLTAGE command

The new **/MEASUREVOLTAGE** command allows ConsoleEDS to measure the Target Voltage (default) on any Equinox programmer. If used with the ISPnano Series of programmers, then this command can also be used to read the '**External Vcc**' input and output voltages.

PC command sent	
ConsoleEDS command	/MEASUREVOLTAGE=Vsource
Command format	ConsoleEDS /MEASUREVOLTAGE= Vsource
Command arguments	The Vsource is the voltage source to be measured: 2 – Target Voltage (default) 3 – External Vcc Input Voltage (ISPnano Series only) 4 – External Vcc Output Voltage (ISPnano Series only)
ConsoleEDS debug output	

4.19 /MEASURECURRENT command

The new **/MEASURECURRENT** command allows ConsoleEDS to measure the current taken by a Target System from the '**Programmer controlled power supply**'.

PC command sent	
ConsoleEDS command	/MEASURECURRENT
Command format	ConsoleEDS /MEASURECURRENT
Command arguments	None
ConsoleEDS debug output	

Please note:

- This command can only be used with the PPM3-MK2, PPM4-MK1 and ISPnano series of programmers.
- The programmer must be controlling power to the Target System for this command to work.

5.0 Creating Projects for ConsoleEDS

5.1 Overview

Any ConsoleEDS command which interacts with a Target IC requires that a so-called '**Base Project**' is specified.

Example:

ConsoleEDS BaseProject.prj /FLASHWRITE=FileName,From,To,FileOffset

Please note:

The Base Project must be a compiled '**Programming Project**' with file extension ***.prj**.

Earlier versions of ConsoleEDS used an uncompiled project with file extension ***.ppm**. This file type is no longer supported by ConsoleEDS as the ***.prj** files now contain additional information about the Target IC allowing ConsoleEDS to be used as a completely standalone application ie without EQTools.

The Base Project tells ConsoleEDS the following information:

- Target IC to be programmed
- Programmer Type eg. Epsilon5
- How to enter Programming Mode (Pre-programming Statemachine settings)
- The Device ID (Signature) of the Target IC
- What voltage to program at (usually the Target Voltage)
- Whether the programmer is to power the Target System (PPM3-MK2 only)
- The SPI or JTAG speed to use
- Whether to Erase the Target IC or not
- The Target IC - Configuration Fuse Settings
- The Target IC - Security Fuse Settings

There are 2 ways of creating a compiled Programming Project as follows:

1. Using EQTools – Project Builder
2. Using EQTools – EDS (Development Mode) and then compiling the project using Project Builder

5.2 Creating a Base Project using EQTools – EDS

5.2.1 Creating an EDS – Development Project

The first step is to create an EDS – Development Project. This project can be used to fully test the interaction with the Target IC BEFORE compiling it for use with ConsoleEDS.

Please follow the steps below to create a Development Project:

- Launch EQTools
 - From the Welcome menu, select <Create a new Equinox Development Project (EDS)>
- or
- From the left-hand pane, select <Development><New Development Project>

→ The Equinox Development Suite (EDS) Wizard will launch.

- Follow the wizard and fill in the relevant information on each tab.
- You can advance to the next page of the Wizard by clicking the <Next> button.

The important information to fill in is as follows:

- Programmer Type
- Target Device (IC)
- Target Power Supply settings

The following tabs in the Wizard are optional:

- Target Oscillator (only used to set the maximum SPI speed)
- FLASH Area – Binary or Hex File (not required for ConsoleEDS if files are to be transferred from the PC via a ConsoleEDS command.)
- EEPROM Area – Binary or Hex File (not required for ConsoleEDS if files are to be transferred from the PC via a ConsoleEDS command.)

At the end of the EDS Wizard, click <Save> and type in a suitable name for your project e.g **myproject.ppm**.

→ The EDS Development Window will now launch

The EDS utility allows you to test out your Programming Project by programming the real Target IC under control of the PC. This can save a lot of development time as it is much easier to find a problem using EDS than using ConsoleEDS.

5.2.2 Testing your EDS – Development Project with the Target IC

The EDS utility allows you to fully test your Development Project with your Target IC / Target System BEFORE compiling it for use with ConsoleEDS.

1. Testing communication with the Target IC

- Select the <FLASH> tab
- Click the <Check Sig> button on the right-hand side
→ EDS will try to read the Device ID (Signature) of the Target IC and display it.
- If EDS reports an error entering Programming Mode, check all the settings are correct for the Device, Power Supply, Statemachine etc.

2. Programming a File into the FLASH of the Target IC

- Select the <FLASH> tab
- Tick the <Edit Buffer> check box
- Select <File Open> and browse to and select your Binary or Hex file
→ the file will be loaded into the Buffer Window
- Click <Write> to program the contents of the Buffer Window in the Target IC FLASH area.

3. Programming a File into the EEPROM of the Target IC

- Select the <FLASH> tab
- Tick the <Edit Buffer> check box
- Select <File Open> and browse to and select your Binary or Hex file
→ the file will be loaded into the Buffer Window
- Click <Write> to program the contents of the Buffer Window in the Target IC FLASH area.

4. Setting the 'Configuration Fuses' for the Target IC

- Select the <Fuses> tab
- Select the required <Fuse Programming Action> (for AVR microcontrollers, always select <Program → RESET → Verify>)
- To read the Fuses from the Target IC, select <Read>
→ the Fuses are read from the Target IC and are displayed in the Target column
- To set the fuses to your desired values, simply click on each fuse in turn and set it to the required value
- To program your Fuse Settings (from the '**PC Fuse State**' column) into the Target IC, click <Write>
→ the Fuse settings in the from the '**PC Fuse State**' column will be programmed into the Target IC and then a verify of these fuse settings will be performed automatically.

Please note:

There are three different places where the '**Configuration Fuses**' can be programmed.

EDS automatically defaults to the correct '**Fuse Tab**' depending on the Target IC selected.

When using ConsoleEDS, you will need to know where the the '**Configuration Fuses**' are stored in the project. This is specified at the top of the EDS <Fuses> tab. E.g. '**Program Post-Erase Fuse Bits**'.

5.2.3 Exporting your EDS Project to Project Builder

Once you have tested your EDS Development Project in EDS, it is then necessary to export it to Project Builder in order to create a compiled Programming Project (*.prj) which can be used by ConsoleEDS.

To export your EDS Project to Project Builder:

- In EDS, select the <Overview> tab
- Click the long button <Open/Modify Base Programming Project>
→ the EDS Window will close
→ your EDS project will now open in '**Project Builder**' view
- On the top icon bar, click the <Compile> icon
→ this will compile your Programming Project (*.ppm) into a *.prj file suitable for use with ConsoleEDS.

If you now use a File Browser to look in your project directory, you should find both a *.ppm and a *.prj file for your Programming Project. The *.prj file is the '**Base Project**' for use with ConsoleEDS.

6.0 Standalone Programming Projects

6.1 Overview

ConsoleEDS allows a pre-compiled '**Standalone Programming Project**' (which is resident in the programmer on-board FLASH Memory) to be executed under the control of a Remote Application. It is possible to have up to 64 '**Programming Projects**' resident in the programmer memory at any point in time. These so called 'Standalone Programming Projects' execute at much higher speed than a similar project executed from the PC as all the FLASH and EEPROM data is stored locally on the programmer.

6.2 Advantages of Standalone Projects

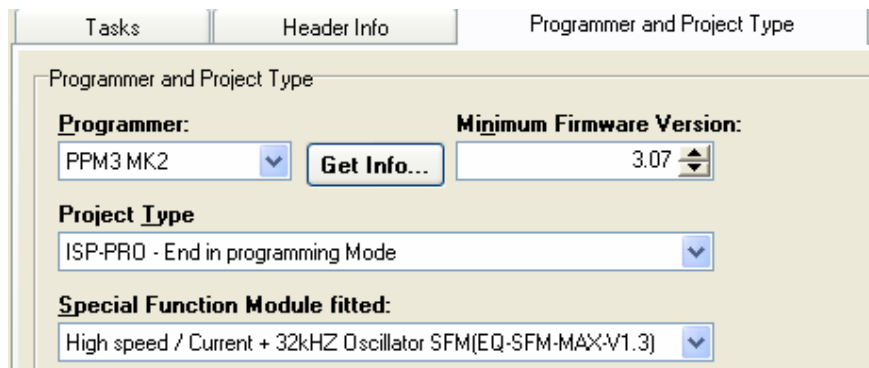
Standalone Projects have the following advantages:

- They contain all programming actions and data in one single Project File including Programming Flags, FLASH CODE File, EEPROM DATA File, Fuse File settings and Security Fuse Settings.
- They are easily version controlled by a single unique '**Project ID**' eg. **ProjectName_1.0.0.1_010207**. This version string applies to the entire Programming Project and so defines all files and parameters for a given product.
- They are much faster than using PC controlled commands as all the data to be programmed is held locally in FLASH memory on the programmer.
- The timings of Standalone Projects are very precise as there are no time delays during the programming procedure due to data being uploaded from the PC.

6.3 Creating a Standalone Project for ConsoleEDS

6.3.1 Defining a ConsoleEDS project

When setting up a Standalone Project for ConsoleEDS, it is necessary to select the "**Project Type**" as "**ISP-PRO – End in programming mode**" – see screenshot below.



These settings mean that once the Standalone Project has executed it will keep the Target System powered up and in “programming mode” so that the next ConsoleEDS command can then be executed. This can save a lot of time in production as powering down and then up again and re-entering programming mode can take up to a second.

6.3.2 Choosing a unique Project Name

Each project must have a unique ‘Project Name’ or ‘Project ID’ which can be used to identify it. It is recommended that ‘version control’ is enabled so that there is no chance of the incorrect project being executed by mistake.

The format of the Unique ID is as follows:

<Project_name> <Date><Version>

Example: **myproject1_150103_2.1.2.22**

Project Name

The Project Name is limited to 16 ASCII characters when version control is enabled.

Only alphanumeric characters should be used as far as possible.

The ‘_’ character is not allowed as this is used as a delimiter. The ‘-’ character can be used instead.

Date

The date is usually the last compilation date of the project.

Version

The version information can be used for whatever version convention is required in your company.

Many customers use the first three parts of the version as the current firmware version and the last part as the build number for the project.

For the example above:

- The project is called ‘myproject1’.
- It was compiled on the 15/01/03.
- The firmware version of the HEX file contained within the project is 2.1.2.
- The project is currently on build 22.

6.3.3 Selecting a Standalone Project

When controlling a programmer using ConsoleEDS, the project must have been set up as a ‘Standalone’ project. This basically means that when the project has finished executing, it will tri-state all I/O pins, power down the Target System (PPM3 MK2 only) and then wait for the next project to be selected.

To set up a ‘Standalone Project’:

- Launch EQTools
- Open your Programming Project using Project Builder
- Select the **<Programmer and Project type>** tab
- Select **‘Standalone – keypad control’** from the drop-down menu
- Compile the project

6.4 Uploading Standalone Projects to a Programmer

6.4.1 Overview

ConsoleEDS now supports uploading of the actual '**Programming Projects**' contained within a '**Project Collection**' to a target programmer directly from the ConsoleEDS command line. This allows a third party application to upload the relevant 'Programming Projects' to the programmer depending on the programming sequence which is to be carried out.

Example:

Let's say you want to upload a Project Collection file (*.ppc) called **ProjectCollection.PPC** which contains 3 Project Files called:

- PROJECT1
- PROJECT2
- PROJECT3

to your attached programmer.

Once the projects have been uploaded to the programmer, you also want to double-check that the correct '**Project Names**' have been uploaded...just in case there is a project name or version mismatch.

Here is the typical sequence of commands to upload the Project Collection, store a cached Project List in a text file and finally to check that the correct projects have been uploaded:

ConsoleEDS /UPLOAD=ProjectCollection.PPC

→ the 'Programming Projects' contained in the **ProjectCollection.PPC** file are uploaded to the attached programmer(s).

ConsoleEDS /READPROJECTSTO=ProjectList.txt

→ ConsoleEDS reads the list of 'Programming Projects' now installed in the programmer back to a text file called **ProjectList.txt**. This cached version of the 'Project List' is then used by all subsequent ConsoleEDS commands to look up a Programming Project.

Optional commands to check whether specified projects have been uploaded:

ConsoleEDS /CHECKPROJECT=PROJECT1 /GetProjectsFrom=ProjectList.txt

→ Checks that '**PROJECT1**' is found in the 'Project List'.

ConsoleEDS /CHECKPROJECT=PROJECT1 /GetProjectsFrom=ProjectList.txt

→ Checks that '**PROJECT2**' is found in the 'Project List'.

ConsoleEDS /CHECKPROJECT=PROJECT1 /GetProjectsFrom=ProjectList.txt

→ Checks that '**PROJECT3**' is found in the 'Project List'.

These commands are described in more detail in the next sections.

6.4.2 Uploading a Project Collection to a Programmer

To upload all projects in a Project Collection to an attached programmer, type the following command:

ConsoleEDS /UPLOAD=ProjectCollection.ppc

Where:

- **ProjectCollection.ppc** is the Project Collection file which contains the Programming Projects to be uploaded to the programmer.

ConsoleEDS will now start to upload the Programming Projects in the Project Collection. The Projects are uploaded and then a separate Verify Pass of each project is performed. ConsoleEDS will generate the following diagnostic text:

Console EDS - Version 2.1 (Build 730)

Current Action: Uploading

Programmer(s): PPM3 MK2 @address:0

Project: (1) -'PROJECT1_1.0.0.1_V130B'

9 % transfer completed

To cancel the Upload process, click the <ESC> key on your PC keyboard.

Current Action: Verifying

Programmer(s): PPM3 MK2 @address:0

Project: (1) -' PROJECT1_1.0.0.1_V130B'

36 % transfer completed

To cancel the Upload process, click the <ESC> key on your PC keyboard.

When the upload of all the Programming Projects is complete, ConsoleEDS will generate either a PASS or FAIL for the Upload Operation:

Here is the ConsoleEDS output of the Upload Operation is successful:

Operation: Project Upload and Verify

Result: PASS

Collection Name: ProjectCollection.ppc

Number of projects: 1

Path:

Uploaded successfully to programmer(s):

Addr	Type	Firmware	Serial No
0	PPM3 MK2	3.02	1066

Closing Console EDS, no errors have occurred.

Here is the ConsoleEDS output if the Upload Operation fails:

ConsoleEDS /UPLOAD=MG02-2.PPC
Console EDS - Version 2.1 (Build 829)
Console EDS - Started at 5/13/2008 1:02:12 PM
Console EDS - Now Uploading and Verifying ProjectCollection.PPC...

Data verify error at byte 0 of Page 28

Operation: Project Upload and Verify
Result: FAIL
Project Name: MG02-2.PPC
Path:

If you receive this error message, it means that one of the pages of the Internal FLASH Memory has failed to program properly. Please contact Equinox for further assistance with this error.

6.4.3 Uploading a Single Project to a Programmer

To upload a single project from a Project Collection to an attached programmer, type the following command:

ConsoleEDS /UPLOAD=ProjectCollection.ppc,ProjectName

Where:

- **ProjectCollection.ppc** is the Project Collection file which contains the Programming Projects to be uploaded to the programmer.
- **ProjectName** is the name (Unique ID) of the Programming Project to be uploaded.

ConsoleEDS will now start to upload the specified Programming Project to the programmer. Once the Project has been uploaded, a separate Verify Pass is performed.

6.5 Downloading a Project List from a Programmer

To download a list of all Programming Projects currently resident in the attached programmer, type the following command:

ConsoleEDS /PROJECTLIST

ConsoleEDS will now output the 'Project Count' followed by a list of all projects in the programmer:

Console EDS - Version 2.1 (Build 730)

Programmer Project List:

Total project count = 5

1: PROJECT1_1.0.0.0

2: CAL-PROG_1.0.0.0

3: CAL-RUN_1.0.0.0

4: CAL-PROG-FAIL_1.0.0.0

5: LP111_1.0.0.1_6712021B

Closing Console EDS, no errors have occurred.

6.6 Downloading a Project List from a Programmer to a file

It is also possible to download a 'Project List' from an attached programmer to a file on the PC hard disk. This can speed up repetitive ConsoleEDS sessions considerably as the action of reading the Project List can be very time-consuming if the programmer contains many projects.

Example:

ConsoleEDS /READPROJECTSTO=ProjectList.txt

→ This command reads the project list from the programmer to the file ***ProjectList.txt***

Any further commands in the ConsoleEDS sequence then simply reference the Project List stored in the ***ProjectList.txt*** file.

Example:

ConsoleEDS BaseProject.prj /AUTOPROGRAM=ProjectName

/GETPROJECTSFROM=ProjectList.txt

→ This command executes the 'Standalone Project' called '***ProjectName***' by looking up the project in the ***ProjectList.txt*** file.

6.7 Checking a Project is resident in the programmer

If a Project Collection (*.ppc) file has been uploaded to a programmer, it is good practice to perform a check to make sure that each 'Standalone Programming Project' which is to be used in a programming sequence has actually been uploaded to the programmer. This prevents errors later on in the sequence.

The **/CHECKPROJECT** is used to check whether a 'Standalone Programming Project' has already been uploaded to a programmer.

Example 1:

ConsoleEDS /CHECKPROJECT=PROJECT1_1.0.0.1

- This command will read the 'List of Standalone Projects' from the programmer and then check whether the project '**PROJECT1_1.0.0.1**' is contained in this list.
- This command can execute quite slowly if there are many projects resident inside the programmer. To speed up the use of this command, the Project List should be cached on a one-time basis using the **/GetProjectsFrom** command.

Example 2:

ConsoleEDS /READPROJECTSTO=ProjectList.txt

ConsoleEDS /CHECKPROJECT=PROJECT1_1.0.0.1 /GetProjectsFrom=ProjectList.txt

- This command sequence will read the Project List to a text file and then will use the cached 'List of Standalone Projects' contained in the **ProjectList.txt** file and will check whether the project '**PROJECT1_1.0.0.1**' is contained in this list.
- This command sequence is much faster as the reading the Project List from the text file is very fast.
- The **/READPROJECTSTO=ProjectList.txt** only needs to be executed once immediately after the **/UPLOAD** command has been executed. All other ConsoleEDS commands can then use the cached 'Project List' stored in the text file.

7.0 Typical ConsoleEDS control sessions

7.1 Typical ConsoleEDS Initialisation

When using ConsoleEDS to control a programmer, it is good practice to initialise the programmer correctly for each programming sequence. This initialisation includes setting the correct COM port, checking the ConsoleEDS build revision and also checking the programmer firmware version. These tasks can be performed using a sequence of commands which are almost identical for each programming sequence.

Here is the typical sequence of ConsoleEDS commands to perform the following tasks:

- Set up the COM Port, BAUD Rate and other communications parameters
- Check the minimum version of ConsoleEDS is build 903
- Check the version of firmware in the attached programmer ≥ 3.07
- Upload the specified Project Collection to the programmer

ConsoleEDS /SETUP=1,0,1000

ConsoleEDS /CHECKMINIMUMBUILD=903

ConsoleEDS /CHECKMINIMUMFIRMWARE=3.07

7.2 Uploading a Project Collection

This section details a typical example of how to upload a set of 'Standalone Projects' (a Project Collection) to a programmer.

Example:

Let's say you want to upload a Project Collection file (*.ppc) called ***ProjectCollection.PPC*** which contains 3 Project Files called:

- PROJECT1
- PROJECT2
- PROJECT3

to an attached programmer.

Once the projects have been uploaded to the programmer, you also want to double-check that the correct '***Project Names***' have been uploaded...just in case there is a project name or version mismatch.

Here is the typical sequence of ConsoleEDS commands to perform the following tasks:

- Upload the specified Project Collection to the programmer
- Read back and store a cached Project List in a text file
- Check that the correct projects have been uploaded to the programmer

ConsoleEDS /UPLOAD=ProjectCollection.PPC

ConsoleEDS /READPROJECTSTO=ProjectList.txt

ConsoleEDS /CHECKPROJECT=PROJECT1 /GetProjectsFrom=ProjectList.txt

ConsoleEDS /CHECKPROJECT=PROJECT2 /GetProjectsFrom=ProjectList.txt

ConsoleEDS /CHECKPROJECT=PROJECT3 /GetProjectsFrom=ProjectList.txt

7.3 Executing a Standalone Project

This section details how to execute (run) a Standalone Project. It is assumed that the project has already been uploaded to the programmer using the /UPLOAD command.

Example:

Let's say you wish to execute the project called '**PROJECT1**'.

The following command would be required:

ConsoleEDS BaseProject.prj /AUTOPROGRAM=PROJECT1 /GetProjectsFrom=ProjectList.txt

Important!

- It is important to use the '**/GetProjectsFrom**' argument for each ConsoleEDS command as this forces each command to use the cached '**Project List**' stored in the file **ProjectList.txt**. This is much faster than ConsoleEDS reading the '**Project List**' from the programmer each time...as this can take up to 5 seconds if there are a lot of projects in the programmer.
- The **BaseProject.prj** is not actually required if you have no other arguments on the command line.

7.4 Writing a BLOCK of data to a specified address range

A typical application of ConsoleEDS is where a BLOCK of data is to be programmed into a Target IC at a specified address range. This is a typical requirement when programming eg. unique serial numbers or calibration data into a IC which will be used by the firmware of the IC during run-time execution.

Here is an example:

Requirement:

A unique 4 byte serial number which is generated by a Remote PC Application (RA) is to be programmed into the FLASH area of the Target IC at address 0x00050 to 0x00053. The FLASH area already contains application code, so this serial number is to be overlayed in the FLASH without erasing the data which is already there.

Implementation:

The simplest way to implement this requirement using ConsoleEDS would be for the RA to write each serial number to a file called eg. serial4.bin. ConsoleEDS will then program this unique serial number into the FLASH of the Target IC starting at the specified address.

This can be achieved with ConsoleEDS using the following command:

ConsoleEDS baseproject.prj /FLASHWRITE=FileName,From,To,FileOffset

Where:

- Baseproject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc.
- FileName.bin is the file name of a BINARY data file
- 'FROM' is the START address to begin programming
- 'To' is the last address to be programmed
- 'FileOffset' is offset address from the start of the file where ConsoleEDS will start to read the data from.

This command below will program the 4 bytes read from the file ***serial4.bin*** into consecutive bytes in the Target IC FLASH starting at address. A Chip Erase is not required as this would erase the data already in the FLASH area.

ConsoleEDS baseproject.prj /FLASHWRITE=serial4.bin, 0x00050, 0x00053,0

7.5 Reading the entire EEPROM area back to a file

In some instances it is necessary to read the entire contents of the Target IC EEPROM back to a file. This may be necessary in order to back up some unique serial number and calibration data which may have already been stored in the EEPROM.

The command to read back the entire EEPROM contents to a file is:

ConsoleEDS baseproject.prj /EEPROMREAD=FileName.bin

Where:

- ***Baseproject.prj*** is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc.
- ***FileName.bin*** is the file name of a BINARY data file

This command will read back the entire contents of the Target IC EEPROM starting at address 0x00000 and ending at the last address of the EEPROM area. The contents will be written to the specified BINARY or Hex File.

7.6 Reading a BLOCK of data from a specified address range back to a file

In many applications it is necessary to read back blocks of data from the EEPROM area of the Target IC so they can be processed / archived by a Remote Application.

Requirement:

A block of 16 bytes of '**Calibration Data**' has been written into the EEPROM of the Target IC in the address range 0x01000 to 0x0100F during the execution of some Test Firmware. This block of data is now to be read back from the Target IC EEPROM to a file.

Implementation

The following ConsoleEDS command would be used to read a block of EEPROM data to a file:

ConsoleEDS baseproject.prj /EEPROMREAD=FileName,AddressFrom,AddressTo

Where:

- **Baseproject.prj** is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc.
- **FileName** is the name of data file into which the data is to be written
- **'AddressFrom'** is the START address in the Target IC from which to start reading data
- **'AddressTo'** is the END address in the Target IC of the Data Block being read

The command below will read a block of data back from the EEPROM address range 0x01000 to 0x01000F to a BINARY file called caldata.bin:

ConsoleEDS baseproject.prj /EEPROMREAD=caldata.bin, 0x01000, 0x0100F

The Remote Application can then read the data from this file and perform any required action upon this data.

7.7 Programming the Configuration Fuses of the Target IC

It is possible to program the '**Configuration Fuses**' of the Target IC by either storing the 'Fuse Values' in a project or by simply passing the Hex value of the fuses on the command line.

7.7.1 Programming Fuses from a Project

If the fuse settings are already stored in a Programming Project, then the following command can be used to program these fuse settings into a Target IC:

ConsoleEDS BaseProject.prj /WRITE_POST_ERASE_FUSES=FuseFile.prj

Where:

- **BaseProject.prj** is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc.

- **FuseFile.prj** is the Programming Project containing the Configuration Fuse Settings to be programmed.

This command will read the fuse settings from the **<Post Erase Fuse>** tab and program them into the Target IC.

Please note:

This command requires a license upgrade for ConsoleEDS PRO.

7.7.2 Programming Fuses using the Hex Fuse Values

If the Hex values of the Fuses are known, then it is possible to simply pass the Fuse Values to be programmed on the Command Line as follows:

ConsoleEDS BaseProject.prj /POSTERASEFUSEWRITE=0x42,0xA9,0xFF

In this case, the Fuse Hex Byte settings are: **0x42,0xA9,0xFF**.

7.8 Programming the Security Fuses of the Target IC

It is possible to program the '**Security Fuses**' of the Target IC by either storing the 'Security Fuse Values' in a project or by simply passing the Hex value of the fuses on the command line.

7.8.1 Programming Fuses from a project

If the 'Security Fuse' settings are already stored in a Programming Project, then the following command can be used to program these fuse settings into a Target IC:

ConsoleEDS BaseProject.prj /WRITESECURITY=SecurityFuseFile.prj

Where:

- BaseProject.prj is a project file on your hard disk containing the settings for this session of ConsoleEDS ie which programmer, IC, voltage etc.
- SecurityFuseFile.prj is the Programming Project containing the Security Fuse Settings to be programmed.

This command will read the Security fuse settings from the **<Security Fuses>** tab and program them into the Target IC.

Please note:

This command requires a license upgrade for ConsoleEDS PRO.

7.8.2 Programming Fuses using the Hex value(s)

If the Hex value(s) of the 'Security Fuses' are known, then it is possible to simply pass the Security Fuse Values to be programmed on the Command Line as follows:

ConsoleEDS BaseProject.prj /WRITESECURITY=SecurityHexValue1, SecurityHexValue2....

Appendix 1 – ConsoleEDS Error Code and Error reporting

1.0 Overview

ConsoleEDS is a '**Console Application**' which can be called / initiated from any other application running within the same session of Windows. This makes **ConsoleEDS** ideal for interfacing with any other control application written in any programming language. As long as so-called '**Remote Application**' can shell to a '**Console Application**' then it should be able to control an Equinox programmer.

It is possible to call / initiate a ConsoleEDS session from any programming language. It is also possible for the '**Remote Application**' to simply run a 'batch file' which contains all the instructions to control ConsoleEDS.

1.1 ConsoleEDS – Exit Code / Error Code

When you call ConsoleEDS from a **Remote Application**, ConsoleEDS will return a so-called "**Exit Code**" when the session ends. This "**Exit Code**" can be used by '**Remote Application**' to determine whether the ConsoleEDS session has passed or failed.

The possible "**Exit Codes**" are as follows:

- If the "**Exit Code**" is 0, then the operation was successful (PASS).
- If the "**Exit Code**" > 0, then the operation failed for some reason. The "**Exit Code**" then represents the "**Error number**" which you can then look up to find out why the operation failed.

The technique of checking the "**Exit Code**" is much simpler and more reliable than parsing the output text from ConsoleEDS for a PASS / FAIL text. Once the result has been determined, the output text can still be parsed if further debug information is required.

1.2 ConsoleEDS – Error Code

When you call ConsoleEDS from a **Remote Application**, ConsoleEDS will return a so-called "**Exit Code**" when the session ends. If this "**Exit Code**" is >0 then the session has ended with an error. The "**Exit Code**" is the "**Error Code**" and can be used to determine why the session failed.

A list of the possible "**Error Codes**" and their meanings can be found in the "**Error messages manual**" which is available to download from the Equinox website. Many of the error codes have slightly different meanings depending on which target device e.g. AVR, AT89S, AT91SAM7 etc is being programmed. It is therefore a good idea to consult the list of errors in the separate application notes for the target device as this may give a better understanding of why the programming operation has failed.

1.3 ConsoleEDS – Batch File - Error Code example

The example below shows how to call ConsoleEDS from a batch file and how to handle an error.

- When the ConsoleEDS session finishes, ConsoleEDS returns '**ERRORLEVEL=0**' if the result was a pass and '**ERRORLEVEL>0**' if the session failed for any reason.
- The batch file checks for an error (non zero value) using the "**IF NOT %ERRORLEVEL%==0 GOTO ERRORx**" command.
- If an error is returned then the batch file jumps to the relevant 'Error tag' in the batch file. e.g. **ERROR1** or **ERROR2**

Here is an example batch file:

```
@ECHO OFF
cls
```

```
rem Run Console EDS to perform the first task e.g get Project List
call ConsoleEDS /PROJECTLIST
IF NOT %ERRORLEVEL%==0 GOTO ERROR1
```

```
rem Run Console EDS to perform the second task, ERASE and write to FLASH
call ConsoleEDS PPM3-ATMega163.PRJ /ERASE "/FLASHWRITE=serial5.bin"
```

```
IF NOT %ERRORLEVEL%==0 GOTO ERROR2
```

```
rem No errors have occurred
@GOTO SUCCESS
```

```
:ERROR1
```

```
echo An ERROR has occurred in retrieving the project list from the programmer
@GOTO END
```

```
:ERROR2
```

```
echo An ERROR has occurred in erasing the device and writing serial.bin file to flash
@GOTO END
```

```
:SUCCESS
```

```
echo Success – The device programmed OK.
@GOTO END
```

```
:END
```

```
Pause
```

Appendix 2 – ConsoleEDS Error Logging

1.0 Overview

It is possible to get **ConsoleEDS** to log all ConsoleEDS sessions which end in error to an “**Error Log Text File**”. This file can then be used to track any problems over a period of time and can be very useful in diagnosing programming issues when programming batches of boards.

1.1 Using the /LOG command

If the **/LOG** command is appended to any ConsoleEDS session, then the entire debug text for that session will be redirected to the specified ‘**Error log file**’ if the session ends with **ErrorLevel >0**.

Here is an example of using the **/LOG** command:

CONSOLEEDS.EXE ATMEL.PRJ /ERASE /LOG=ProgrammingErrorLog.txt

It is necessary to use the **/LOG** command at the end of each ConsoleEDS session.

1.2 Using the /SETSTARTUPLOG command

If you are using ConsoleEDS build 2001 or higher, then it is possible to set up the ‘**Error Log File**’ once at the start of your ConsoleEDS initialisation.

Here is an example of using the **/SETSTARTUPLOG** command:

ConsoleEDS /SETSTARTUPLOG=ErrorLog.txt

This command stores the setting in your PC registry. All ConsoleEDS commands executed after this will automatically log any error / debug information to the specified file.

Appendix 3 - Programmer Error numbers

The table below details all possible error codes / messages which an Equinox programmer can generate. Any error code numbers which are not specified here are reserved for future use. For a more in-depth description of each error and also suggestions on how to rectify the problem, please refer to relevant application note for the target device being programmed.

Table 1 – Error messages by error number (short-form version)

Error Code	Error message	Error Description
0	NO ERROR	No Error – ie. operation completed successfully
1-9	Internal Errors	Please contact Equinox if you received any of these errors
10	REBOOTED	Programmer has rebooted.
11	Programmer controlled Power Supply - VOLTAGE FAIL	The programmer has tried to setup the programmer controlled power supply, but the voltage has not reached the specified voltage within the power supply timeout specified.
12	VOUT FET FAIL	Programmer could not switch on Target Vcc Output Transistor
13	CRC CHECK FAILED	Programmer Internal FLASH CRC Check Failed
14	PROG CONFIGURATION FAIL	Project Collection stored in Internal FLASH in corrupt or invalid
15	OPERATION ABORTED	The current operation was aborted.
16	PROG_COUNT_EXCEEDED	The number of 'Programming Credits' for the specified project was exceeded.
17	DEVICE ERASE TIMEOUT	A timeout for the Chip Erase operation has elapsed. (Atmel AT91SAM7, T89C51Rx2 and Philips P89CRx2 / 66x devices only.)
18-35	Internal Errors	Please contact Equinox if you received any of these errors
36	INT FLASH PROG FAIL	Failed to program the programmer Internal FLASH Memory
37	INT FLASH VER FAIL	Failed to verify the programmer Internal FLASH Memory
38	INT FLASH TIMEOUT	Internal FLASH timed out during when accessed.
39	INT_FLASH_BAD_PAGE_ALIGNMENT	The uploaded project is set to '512 Byte Pages' but it does not start on a programmer 512 Byte page boundary.
40	CAN NOT ENABLE PROGRAMMING MODE	Failed to enter 'Programming Mode' <ul style="list-style-type: none"> • (AVR – SPI Programming Algorithm) • Failed to enter 'JTAG Debug Mode' (SAM7) • Failed to enter 'JTAG Mode' (AVR)

Error Code	Error message	Error Description
41	SIGNATURE or JTAG ID MISMATCH	The ' Signature ' (Device ID / Chip ID) and / or ' JTAG ID ' read from the Target Device does not match any of the signatures specified in the EQTools project.
42	INVALID MICRO/DEVICE	The microcontroller / device specified in the EQTools project is not supported by the current version of programmer firmware.
43	TARGET DEVICE IS LOCKED	The Target Device is reporting the 'Locked' Device Signature (ID) when the Security Fuses are read back after an 'Erase' operation.
44	PRE-ERASE FUSE PROGRAM / VERIFY FAIL	Failed to program / verify the ' Fuse Bits ' before performing an Erase Operation
45	POST-ERASE FUSE PROGRAM FAIL	Failed to program / verify the ' Fuse Bits ' after performing an Erase Operation
46	FLASH BLANK FAIL	Target Device FLASH Memory is not blank.
47	FLASH PROGRAM FAIL	Failed to program a byte/word/page of FLASH in the Target Device
48	EEPROM BLANK FAIL	EEPROM is not BLANK
49	EEPROM PROGRAM FAIL	Failed to program a location in the EEPROM area
50	PRE-LOCK FUSE PROGRAM FAIL	Failed to program 'Pre-Lock' fuses
51	LOCK PROGRAM VERIFY FAIL	Failed to program/verify Lock Bits
52	OSCAL PROGAM FAIL	Failed to program AVR ' Oscillator Calibration Byte '
53	OSCAL VERIFY FAIL	Failed to verify AVR ' Oscillator Calibration Byte '
54	OUTPUT CURRENT FAIL	The current supplied from the programmer to the Target System was measured and found to be not within the limits specified in the EQTools – Programming Project.
55	BUFFER SUPPLY IS OUTSIDE LIMITS	The voltage applied to the programmer 'Buffer Driver Circuitry' is outside of the voltage limits specified in the EQTools – Programming Project.
56	FLASH VERIFY FAIL	A 'FLASH Verify' operation of the FLASH area of the target device has failed.
57	EEPROM VERIFY FAIL	An 'EEPROM Verify' operation of the EEPROM area of the target device has failed.
58	TARGET RUN TIMEOUT	The 'Target Run' operation timeout has elapsed.

59	PROJECT DOES NOT EXIST IN PROGRAMMER	Either ISP-PRO or ConsoleEDS have tried to execute a Programming Project but the name (Unique ID) of this project cannot be found in the Programmer FLASH Memory Store.
60	INSUFFICIENT_DATA	There is insufficient data for the requested programmer operation.
61	INCORRECT_DATA	Too much data was sent for the requested programmer operation.
62	MISP_FUSE_PROGRAM_FAIL	The PC-controlled ' Fuse Write ' operation has failed for some reason.
63	PSU_DISCHARGE_TIME_EXCEEDED_FAIL	This error message is specific to the ISPnano Series 2 and 3 programmers which feature a ' Target Discharge Circuit ' which is controlled by the programmer. If the Target Vcc has not decayed in the specified time then this error will be displayed.
64	EXT_PSU_OUTSIDE_LIMITS	The voltage connected to the ISPnano 'EXT-VCC' connector is outside of the limits set in the project. Feature not implemented yet.
65	AUTO_CAL_DATA_INVALID	This error is related to a problem with Eqtools sending autocal data when calibrating the power supply of a PPM4-MK1 or ISPnano programmer.
66-255	RESERVED FOR FUTURE USE	Error code is not currently implemented but may be used in the future.