Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

Phyton™

CPI2-B1 In-System Device Programmer

User's Guide

Member of ChipProg-ISP2 family

© 2017 Phyton, Inc. Microsystems and Development Tools

CPI2-B1 In-System Device Programmer

© 2017 Phyton, Inc. Microsystems and Development Tools

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: August 2017 in (whereever you are located)

Table of Contents

Foreword 0

Part I Introduction 17

1 Terminology 17

2 CPI2-B1 device programmer 19

- Features Overview 19
- Hardware characteristics 20
- Software features 21
- Connector TARGET 22
- Connector CONTROL 23
- Single- and Gang-programming control modes 25

Part II Installation and Launching 27

1 Getting Assistance 27

2 Hardware installation 28

3 System Requirements 30

4 Software Installation 30

5 Startup Dialog 34

6 Launching device programmers 36

Part III Control Interfaces 38

1 Using Projects 39

2 Graphical User Interface 40

- User Interface Overview 40
- Toolbars 41
- Menus 42
  - The File Menu 43
    - Configuration Files 44
  - The View Menu 44
  - The Project Menu 44
    - The Project Options Dialog 45
    - The Open Project Dialog 46
    - Export and Import Project Dialogs 46
    - Project Repository 49
  - The Configure Menu 50
    - The Select Device dialog 50
    - The Buffers dialog 52
      - The Buffer Configuration dialog 53
    - The Serialization, Checksum, and Log Dialog 55
      - Shadow Areas 56
      - General settings 57
      - Device Serialization 57
      - Checksum 58
      - Signature string 59
      - Custom Shadow Areas 60

Log file ... 60
The Preferences Dialog ... 61
The Environment Dialog ... 63
Fonts ... 64
Colors ... 64
Mapping Hot Keys ... 65
Toolbar ... 66
Messages ... 66
Miscellaneous Settings ... 66
The Editor Otions Dialog ... 67
The General Tab ... 67
The Key Mappings Tab ... 69
The Edit Key Command Dialog ... 69
The Commands Menu ... 70
Calculator ... 71
The Script Menu ... 72
The Window Menu ... 73
The Help Menu ... 73
License Management Dialog ... 74
Windows ... 76
The Device Information Window ... 76
The Device and Algorithm Parameters Window ... 77
The Buffer Dump Window ... 79
The 'Configuring a Buffer' dialog ... 81
The 'Buffer Setup' dialog ... 82
The 'Display from address' dialog ... 84
The 'Modify Data' dialog ... 84
The 'Memory Blocks' dialog ... 84
The 'Load File' dialog ... 86
File Formats ... 87
The 'Save File' dialog ... 88
The Console Window ... 89
The Program Manager Window ... 89
The Program Manager tab ... 90
Auto Programming ... 91
The Options tab ... 92
Split data ... 92
The Statistics tab ... 94
The Memory Card Window ... 95
Windows for Scripts ... 96
3 Simplified User Interface ... 96
Settings of Simplified User Interface ... 99
Operations with Simplified User Interface ... 103
4 Command Line Interface ... 103
Command Line Options ... 104
5 On-the-Fly Control Interface ... 108
On-the-Fly Command Line Options ... 109
On-the-Fly utility return codes ... 113
On-the-Fly Control Examples ... 114
Part IV Operating Procedures 115
1 How to check if device is blank ... 115

2 How to erase a device 115
3 How to read data from device 115
4 How to program a device 116
How to load a file into a buffer 116
How to edit data before programming 116
How to configure target device 116
How to write information into the device 116
5 How to verify programming 117
6 How to save data to disk 117
7 Multi-Target Programming 118

Part V Integration with NI LabVIEW 118
1 LabVIEW Integration Using Command Line 119
2 LabVIEW Integration Using ACI 122

Part VI Standalone Operation Mode 125
1 Overview 125
2 Switching to and from Standalone Mode 125
3 Preparing Standalone Mode Projects 127
Data Caching 127
Projects and Jobs 129
Device serialization 129
Permissions and setting limits 132
4 Standalone Mode Monitor 133
5 Example of Setting Up Standalone Mode 135

Part VII Software Development Kit (SDK) 146
1 ACI Components 146
2 Using ACI 147
3 Controlling Multiple Programmers via ACI 148
4 ACI Functions 148
5 ACI Structures 151
6 Examples 152
7 API Explorer 154

Part VIII Scripting 156
1 Scripting Overview 156
Simple example 156
2 The Startup Script 157
3 Running Scripts 157
The Script Files Dialog 158
The User Window 160
The I/O Stream Window 160
4 Debugging a Script 160

The Script Window ... 161
Menu and Toolbar ... 161
The AutoWatches Pane ... 162
The Watches Window ... 162
The Display Watches Options Dialog ... 163
The Add Watch Dialog ... 164
5 Script Editor ... 164
The File Menu ... 166
The Edit Menu ... 166
Block Operations ... 167
Condensed Mode ... 168
Syntax Highlighting ... 169
Automatic Word Completion ... 169
The Quick Watch Function ... 170
Dialogs ... 170
The Search for Text Dialog ... 170
The Replace Text Dialog ... 171
The Confirm Replace Dialog ... 172
The Multi-File Search Results Dialog ... 172
Search for Regular Expressions ... 173
The Set/Retrieve Bookmark Dialogs ... 173
The Condensed Mode Setup Dialog ... 174
The Display from Line Number Dialog ... 174

Part IX Reference 174

1 Error Messages ... 174
Error Load/ Save File ... 174
Error Addresses ... 175
Error sizes ... 175
Error command-line option ... 175
Error Programming option ... 176
Error DLL ... 176
Error USB ... 176
Error programmer hardware ... 177
Error internal ... 177
Error confiquration ... 177
Error device ... 177
Error check box ... 178
Error mix ... 178
Warning ... 178
2 Expressions ... 178
Operations ... 179
Operands ... 180
Expression Examples ... 181
3 Scripting Reference ... 181
Scripting Language Description ... 181
Difference Betw een Scripting and C Languages ... 181
Scripting Language Syntax ... 183
Format ... 183
Comments ... 183
Identifiers ... 183
Reserved w ords ... 184

Integer constants ... 184
Long integer constants ... 184
Floating-point constants ... 185
Character constants ... 185
String constants ... 186
Basic Data Types ... 186
Data byte order ... 186
Operations and Expressions ... 187
Operand Metadesignation ... 187
Arithmetic Operations ... 188
Assignment Operations ... 189
Relation Operations ... 191
Logical Operations ... 191
Array Operations ... 192
Bit Operations ... 192
Other Operations ... 194
Operation Execution Priorities and Order ... 194
Operand Execution Order ... 195
Arithmetic Conversions in Expressions ... 196
Operators ... 196
Format and nesting ... 197
Operator label ... 197
Composite operator ... 197
Operator-expression ... 197
Operator Break ... 198
Operator Continue ... 198
Operator Return ... 199
Operator Goto ... 199
Conditional Operator If-Else ... 199
Cycle Operator While ... 200
Cycle Operator Do-While ... 201
Cycle Operator For ... 201
Functions ... 202
Function Definition ... 202
Function Call ... 203
The main Function ... 203
Descriptions ... 203
Basic Types ... 204
Arrays ... 204
Local Variable Definition ... 204
Global Variable Definition ... 205
Variable Initialization ... 205
External Object Description ... 206
Directives of the Script Language Preprocessor ... 206
Identifier Change (#define) ... 207
Inclusion of Files (#include) ... 207
Conditional Compilation ... 207
Predefined Symbols in the Script File Compilation ... 208
Built-in Functions by Group ... 208
Buffer access functions ... 209
CheckSum ... 209
GetByte ... 210
GetDword ... 210
GetMemory ... 210

GetWord ... 211
LoadProgram ... 211
MaxAddr ... 211
MinAddr ... 212
ReloadProgram ... 212
SaveData ... 212
SetByte ... 213
SetDevice ... 213
SetDword ... 213
SetMemory ... 214
SetWord ... 214
Device programming control functions and variables ... 214
Function AllProgOptionsDefault ... 215
Function ExecFunction ... 215
Function GangExecute ... 216
Function GangGetError ... 216
Function GangStatus ... 216
Function GangWaitComplete ... 216
Function GetBadDeviceCount ... 217
Function GetGoodDeviceCount ... 217
Function GetProgOptionBits ... 217
Function GetProgOptionFloat ... 217
Function GetProgOptionList ... 217
Function GetProgOptionLong ... 218
Function GetProgOptionString ... 218
Function mprintf ... 218
Function OpenProject ... 218
Function ProgOptionDefault ... 218
Function ReadShadow Area ... 218
Function SetProgOption ... 219
Function WriteShadow Area ... 219
Variable BlankCheck ... 220
Variable BufferStartAddr ... 220
Variable Checksum ... 220
Variable ChipEndAddr ... 220
Variable ChipStartAddr ... 220
Variable DeviceBatchSize ... 220
Variable DialogOnError ... 221
Variable GangMode ... 221
Variable InsertTest ... 221
Variable LastErrorMessage[] ... 221
Variable NumSites ... 221
Variable ReverseBytesOrder ... 221
Variable SerialNumber ... 221
Variable Signature ... 222
Variable VerifyAfterProgram ... 222
Variable VerifyAfterRead ... 222
Mathematical functions ... 222
String operation functions ... 223
Character operation functions ... 224
Functions for file and directory operation ... 225
Stream file functions ... 226
Formatted input-output functions ... 227
Script File Manipulation Functions ... 227

Text editor functions 228
Debug shell control functions 229
Window s operation functions and other system functions 230
Graphical output functions 231
I/O Stream w indow operation functions 232
Event Wait Functions 232
Other Various Functions 233
Built-in Variables by Group 233
List of Built-in Functions and Variables 234
Scripting Functions 241
fnmerge 241
Function _ff_attrib 242
Function _ff_date 242
Function _ff_name 242
Function _ff_size 242
Function _ff_time 243
Function _fullpath 243
Function _GetWord 243
Function _printfv 243
Function abs 244
Function acos 244
Function ActivateWindow 244
Function AddButton 244
Function AddrExpr 245
Function AddWatch 245
Function API 245
Function asin 246
Function atan 246
Function atof 246
Function atoi 246
Function BackSpace 247
Function BlockBegin 247
Function BlockCopy 247
Function BlockDelete 247
Function BlockEnd 247
Function BlockFastCopy 248
Function BlockMove 248
Function BlockOff 248
Function BlockPaste 248
Function CallLibraryFunction 248
Function ceil 248
Function chdir 249
Function CheckSum 249
Function chsize 249
Function ClearAllBreaks 250
Function ClearBreak 250
Function ClearBreaksRange 250
Function clearerr 250
Function ClearWindow 251
Function close 251
Function CloseProject 251
Function CloseWindow 251
Function cos 251
Function Cr 252

Function creat 252
Function creatnew 252
Function creattemp 253
Function CurChar 254
Function Curcuit 254
Function delay 254
Function DelChar 254
Function DelLine 255
Function difftime 255
Function DisplayText 255
Function DisplayTextF 256
Function Down 256
Function dup 256
Function dup2 256
Function Ellipse 257
Function eof 257
Function Eof 257
Function Eol 258
Function exec 258
Function ExecMenu 258
Function ExecScript 259
Function exit 259
Function ExitProgram 260
Function exp 260
Function Expr 260
Function fabs 260
Function fclose 260
Function fdopen 261
Function feof 262
Function ferror 262
Function fflush 262
Function fgetc 262
Function fgets 263
Function FileChanged 263
Function filelength 263
Function fileno 263
Function FillRect 264
Function findfirst 264
Function findnext 264
Function FindWindow 265
Function FirstWord 265
Function FloatExpr 265
Function floor 265
Function fmod 266
Function fnsplit 266
Function fopen 266
Function ForwardTill 267
Function ForwardTillNot 267
Function fprintf 267
Function fputc 268
Function fputs 268
Function FrameRect 268
Function fread 269
Function FreeLibrary 269

Function freopen 269
Function frexp 270
Function fscanf 270
Function fseek 271
Function ftell 271
Function fw rite 272
Function GetByte 272
Function getc 272
Function getcurdir 273
Function getcw d 273
Function getdate 273
Function getdfree 274
Function getdisk() 274
Function getenv 274
Function GetFileName 274
Function getftime 275
Function GetLine 275
Function GetMark 275
Function GetMemory 276
Function GetScriptFileName 276
Function gettime 276
Function getw 277
Function GetWindow Height 277
Function GetWindow Width 277
Function GetWord 278
Function GetWord 278
Function GotoXY 278
Function HStep 278
Function inport 279
Function inportb 279
Function Inspect 279
Function InvertRect 279
Function isalnum 280
Function isalpha 280
Function isascii 280
Function isatty 280
Function iscntrl 281
Function isdigit 281
Function isgraph 281
Function islow er 281
Function isprint 282
Function ispunct 282
Function isspace 282
Function isupper 282
Function isxdigit 283
Function itoa 283
Function LastChar 283
Function LastEvent 283
Function LastEventInt{1...4} 284
Function LastString 284
Function LineTo 284
Function LoadDesktop 285
Function Left 285
Function LoadLibrary 285

Function LoadOptions ... 285
Function LoadProgram ... 285
Function LoadProject ... 286
Function locking ... 286
Function log ... 287
Function log10 ... 287
Function lseek ... 287
Function ltoa ... 288
Function MaxAddr ... 288
Function memccpy ... 288
Function memchr ... 288
Function memcmp ... 289
Function memcpy ... 289
Function memicmp ... 289
Function memmove ... 290
Function memset ... 290
Function MessageBox ... 290
Function MessageBoxEx ... 291
Function MinAddr ... 291
Function mkdir ... 291
Function MoveTo ... 292
Function MoveWindow ... 292
Function movmem ... 292
Function open ... 293
Function OpenEditorWindow ... 293
Function OpenStreamWindow ... 294
Function OpenUserWindow ... 294
Function OpenWindow ... 294
Function outport ... 295
Function outportb ... 295
Function peek ... 295
Function peekb ... 295
Function poke ... 296
Function pokeb ... 296
Function Polyline ... 296
Function pow ... 296
Function pow10 ... 297
Function printf ... 297
  printf Conversion Type Characters ... 298
  printf Flag Characters ... 299
  printf Format Specifier Conventions ... 299
    %e or %E Conversions ... 299
    %f Conversions ... 299
    %g or %G Conversions ... 300
    %x or %X Conversions ... 300
    Alternate Forms for printf Conversion ... 300
  printf Format Specifiers ... 300
  printf Format String ... 301
  printf Input-size Modifiers ... 301
  printf Precision Specifiers ... 302
  printf Width Specifiers ... 303
Function pscanf ... 303
Function putc ... 304
Function putenv ... 305

Function putw ... 305
Function rand ... 305
Function random ... 305
Function randomize ... 305
Function read ... 306
Function Rectangle ... 306
Function Redraw Screen ... 306
Function ReloadProgram ... 307
Function RemoveButtons ... 307
Function rename ... 307
Function rew ind ... 307
Function Right ... 308
Function rmdir ... 308
Function SaveData ... 308
Function SaveDesktop ... 309
Function SaveFile ... 309
Function SaveOptions ... 309
Function scanf ... 309
Function Search ... 310
Function searchpath ... 311
Function SearchReplace ... 311
Function SelectBrush ... 311
Function SelectFont ... 311
Function SelectPen ... 312
Function SetBkColor ... 312
Function SetBkMode ... 312
Function SetBreak ... 313
Function SetBreaksRange ... 313
Function SetByte ... 313
Function SetCaption ... 313
Function setdisk ... 313
Function SetDw ord ... 314
Function SetFileName ... 314
Function setftime ... 314
Function SetMark ... 315
Function setmem ... 315
Function SetMemory ... 315
Function setmode ... 315
Function SetPixel ... 316
Function SetTextColor ... 316
Function SetToolbar ... 316
Function SetUpdateMode ... 316
Function SetWindow Font ... 317
Function SetWindow Size ... 317
Function SetWindow SizeT ... 318
Function SetWord ... 318
Function sin ... 318
Function sprintf ... 318
Function sqrt ... 319
Function srand ... 319
Function sscanf ... 319
Function Step ... 320
Function Stop ... 320
Function stpcpy ... 320

© 2017 Phyton, Inc. Microsystems and Development Tools

Function strcat 320
Function strchr 321
Function strcmp 321
Function strcmpi 321
Function strcpy 321
Function strcspn 322
Function stricmp 322
Function strlen 322
Function strlw r 322
Function strncat 323
Function strncmp 323
Function strncmpi 323
Function strncpy 324
Function strnicmp 324
Function strnset 324
Function strpbrk 324
Function strrchr 325
Function strrev 325
Function strset 325
Function strspn 325
Function strstr 326
Function strtol 326
Function strtoul 327
Function strupr 327
Function tan 327
Function tanh 327
Function tell 328
Function TerminateAllScripts 328
Function TerminateScript 328
Function Text 328
Function toascii 329
Function Tof 329
Function tolow er 329
Function toupper 329
Function ultoa 329
Function unlink 330
Function unlock 330
Function Up 330
Function UpdateWindow 331
Function Wait 331
Function WaitExprChange 331
Function WaitExprTrue 332
Function WaitGetMessage 332
Function WaitMemoryAccess 332
Function WaitSendMessage 333
Function WaitStop 334
Function WaitWindow Event 334
Function w getchar 335
Function w gethex 335
Function w getstring 336
Function Window Hotkey 336
Function WordLeft 336
Function WordRight 336
Function w printf 336

Function w rite 337
lock 337
Variable _fmode 338
Variable ApplName 338
Variable BlockCol1 338
Variable BlockCol2 338
Variable BlockLine1 338
Variable BlockLine2 339
Variable BlockStatus 339
Variable CaseSensitive 339
Variable CurCol 339
Variable CurLine 339
Variable DesktopName 340
Variable errno 340
Variable InsertMode 340
Variable LastFoundString 340
Variable LastMemAccAddr 340
Variable LastMemAccAddrSpace 341
Variable LastMemAccLen 341
Variable LastMemAccType 341
Variable LastMessageInt 341
Variable LastMessageLong 341
Variable MainWindow Handle 341
Variable NumWindow s 342
Variable RegularExpressions 342
Variable SelectedString 342
Variable SystemDir 342
Variable WholeWords 342
Variable Window Handles 343
Variable WorkFieldHeight 343
Variable WorkFieldWidth 343

4 ACI Fuctions and Structures 343

ACI Fuctions 343

ACI_Launch 343
ACI_Exit 343
ACI_ErrorString 344
ACI_LoadConfigFile 344
ACI_SaveConfigFile 344
ACI_LoadProject 345
ACI_SetDevice 345
ACI_GetDevice 345
ACI_GetLayer 346
ACI_CreateBuffer 346
ACI_ReallocBuffer 346
ACI_ReadLayer 346
ACI_WriteLayer 347
ACI_FillLayer 347
ACI_GetProgrammingParams 347
ACI_SetProgrammingParams 348
ACI_GetProgOption 348
ACI_SetProgOption 349
ACI_AllProgOptionsDefault 349
ACI_ExecFunction 350
ACI_StartFunction 350

ACI_GangStart 350
ACI_GetStatus 351
ACI_TerminateFunction 351
ACI_GangTerminateFunction 351
ACI_FileLoad 351
ACI_FileSave 352
ACI_SettingsDialog 352
ACI_SelectDeviceDialog 352
ACI_BuffersDialog 352
ACI_LoadFileDialog 354
ACI_SaveFileDialog 354
ACI_SerializationDialog 355
ACI_SetConnection 355
ACI_GetConnection 356
ACI_ConnectionStatus 356

ACI Structures 356

ACI_Launch_Params 356
ACI_ErrorString_Params 357
ACI_Buffer_Params 357
ACI_Config_Params 359
ACI_ProjectParams 360
ACI_Connection_Params 360
ACI_Device_Params 360
ACI_File_Params 360
ACI_Function_Params 362
ACI_GangStart_Params 363
ACI_GangTerminate_Params 364
ACI_Layer_Params 364
ACI_Memory_Params 366
ACI_ProgOption_Params 366
ACI_Programming_Params 371
ACI_PStatus_Params 373

Index 376

1 Introduction

Phyton™

CPI2-B1
In-System Device Programmers

User's Guide

Copyright © 2017, Phyton, Inc. Microsystems and Development Tools, All rights reserved

1.1 Terminology

Terms used in the document

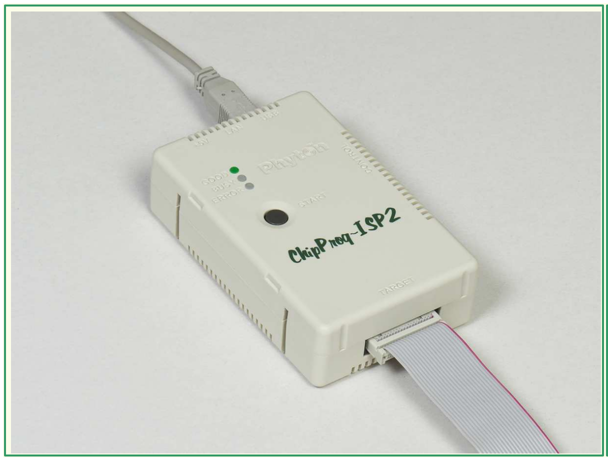
ISP or **in-system programming**	Operations on device mounted on a board in user equipment. ICP is performed via a cable connecting programmer to the target either directly or via needles or pogo contacts.
ICP or **in-circuit programming**	Same as ISP above.
	Mode of the in-system programming that is usually defined by the programming signals voltage or the ISP interface (JTAG, SWD, UART, SPI, etc.). Distinct ISP modes are enabled for different target devices and more than one mode may exist for one device.
Target device or **Target**	A serial flash memory device, microcontroller or programmable logical device having memory inside which can be programmed by an in-system device programmer. In CPI2-B1 GUI device names comprised of part numbers (full or reduced) following types of ISP programming modes in [ ] brackets (for example: PIC10F200 [ISP HV Mode], M25PX80 [ISP Mode]).
DUT	Device Under Test - same as target device above.
Start and **End Addresses** (of the Target device)	Physical memory range of target device to perform programming operations (read, write, verify, etc.) on.
Programming Interface	On-device port that enables access to the internal memory that includes but not limited to: SPI, I2C, JTAG, SWD, UART.
ISP Mode	Mode of the in-system programming. Distinct ISP modes are enabled for

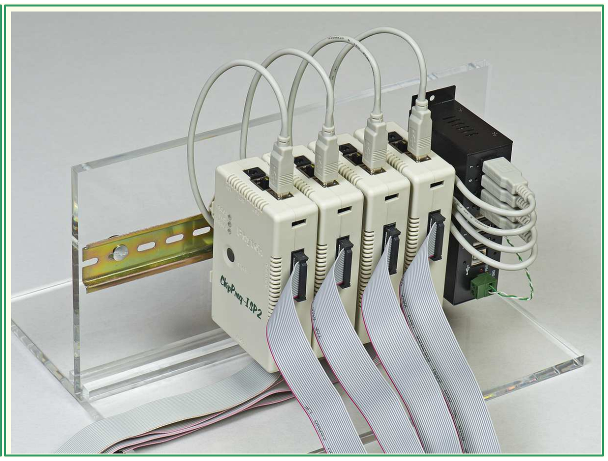
	different target devices and more than one mode may exist for one device.
ISP JTAG Mode	In-system programming using JTAG interface.
ISP SWD Mode	In-system programming using SWD (single wire debug) interface.
ISP EzPort Mode	In-system programming using Freescale proprietary EzPort interface.
ISP HV Mode	In-system programming that requires application of relatively high voltage to the target device (12V for example).
File	In the CPI2-B1 context the term **file** may represent: a) an image of information on a PC hard drive or other media that is supposed to be written into the target device's physical memory, or b) an image fetched from the target device and stored on the disk or other media. Files in ChipProg can be read from and writted to a PC hard drive or CD.
Buffer or **Memory buffer**	Buffers are intermediate data holders between data in files and data in the target device. A buffer is a portion of computer memory (RAM) used to temporarily store, edit and display data to be written to the target device or read from the device. User can open any number of buffers of any size only limited by available computer memory.
Buffer layer or **sub-layer**	A buffer may hold several layers (also known as sub-layers) that according to architecture and memory model of a particular target device. For example, for some microcontrollers one buffer can include the code and data memory layers (see more details below).
Buffer size	Buffers size may vary from 128KB to 32GB.
Buffer start address	The address to display the buffer contents from.
Checksum	An arithmetic sum of all bytes of data in a specified part of buffer calculated by programmer to ensure data integrity. The program has a variety of algorithms for checksum calculation and allows writing the checksum into a specified location of the target device.
Command Line mode	Method of controlling a CPI2-B1 in which the user issues commands to the computer program in the form of successive lines of text (command lines).
Standalone Operation Mode	CPI2-B1 device programmer contains internal memory card that can hold all information that the device programmer needs to run without further interaction with a PC.
Project	An integrated set of information that completely describes the target device, properties of data buffers, programming options and settings, list of source and destination files with their properties, etc. Each project with a unique name can be stored and promptly reloaded for immediate execution. Usually user creates a project to work with one type of device. Using projects saves a lot of time during initial configuration of programmer every time you start working with a new device.

1.2 CPI2-B1 device programmer

ChipProg-ISP2 is a family of in-system device programmers produced by Phyton, Inc. Microsystems and Development Tools. This family currently represented by two models: a single-channel CPI2-B1 and CPI2-Gx gang device programmer.

CPI2-B1 device programmers are primarily intended for use in test fixtures for programming single boards and multi-board panels. For this purpose multiple CPI2-B1 units can be driven from one computer in the gang mode. This device programmer can be also used for engineering and field service. The programmer works under control of the ChipProg-02 software package. See a single CPI2-B1 and four CPI2-B1 units mounted on a rail on the pictures below.





1.2.1 Features Overview

Features Overview

- Programs devices with Vcc from 1.2V to 5.5V.
- Supports JTAG, SWD, SPI, SCI, I²C, UART, and other interfaces.
- Extremely fast.
- Can program some devices at a long distance of up to 5m (~15ft).
- Up to 72x CPI2-B1 units can be controlled by a single computer.
- Each of ganged programmers works independently.
- USB 2.0 High Speed and LAN 100 Mbit/s communication interfaces.
- Opto-isolated RS-232 interface (optional).
- ATE interface for stand-alone operations.
- Each module has memory card that enables stand-alone operations.
- Friendly intuitive graphical user interface (GUI).

- Simplified graphical user interface for use by unskilled personnel.
- Application Control Interface (ACI) provided by a DLL.
- ACI enables control from programs in Visual Basic, C, C++, C#, etc.
- ACI enables control from National Instrument® LabVIEW™.
- On-the-fly utility allows controlling already launched programmer.
- Software includes scripting language.
- Project files are protected against hackers and corruption.
- Programmer kit includes a bracket for mounting on a standard DIN rail.
- Clip-on compartment for a battery, LEDs and a button for standalone operations (optional).

1.2.2 Hardware characteristics

NOTE. Some of the features and items below may be unavailable by the moment of sale of your CPI2-B1 device programmer

Housing Options and Applications
- Palm-size unit in a plastic enclosure.
- User-configurable gang programming system comprised of single CPI2-B1 units mounted on a standard DIN rail
- Hand-held battery-powered tool for in-field service.

Extra Options and Ordering Codes
- CPI2-B1 – single-channel programmer with no galvanic isolation of control lines.
- CPI2-ISO – single-channel programmer with galvanic isolation of control lines and RS-232 interface.
- CPI2-BB – add-on compartment with Li-Ion battery and controls for stand-alone operation.
- All above options include plastic brackets for mounting programmer units on a standard EN 50022 (TS35) 35 mm DIN rail.

Communication interfaces
- USB 2.0 High-speed.
- 100 Mbit/s Ethernet (LAN).
- RS-232C (with CPI2-ISO option only).

Powering the programmer
- From external power supply 5V/1A (not included).
- From PC USB port.
- Rechargeable Li-Ion battery (with CPI2-BB option only).

Powering Targets from the Programmer
- When powered from an external power supply (5V@1A), provides the target equipment with the voltages: Vcc (1.2 to 5.5V @ up to 350mA) and Vpp (1.2 to 15V @ up to 80mA).

Signals to/from the Target
- Ten input/output lines with logical levels 1.2 to 5.5V that can be individually programmed as TTL/ CMOS logic I/O.
- The signal lines above alternate with GND lines for stable programming via long cables.
- Two input/output lines which can be individually programmed as TTL logic I/Os, GNDs, Vcc or Vpp.

Control Methods
- Start/Stop logic signal for external control.
- Output signals for external control: BUSY, GOOD and ERROR.
- Six logic inputs for choosing one of 64 preloaded projects.
- One low-current output for setting that can be used for project selection code.

- One output signal for charging an add-on battery (CPI2-BB).
- Three GND lines.

Dimensions
- CPI2-B1 unit: 114 x 73 x 32 mm (~4.5 x 2.9 x 1.25 inch).
- With CPI2-BB battery: 114 x 99 x 32 mm (~4.5 x 3.9 x 1.25 inch).

1.2.3 Software features

NOTE. Some of the features and items below may be unavailable by the moment of sale of your CPI2-B1 device programmer.

System Requirements
- Microsoft® Windows™ 7, 8 or 10.

Software Features
- Supports loading and saving files in all popular formats.
- Unlimited number of data buffers can be open and maintained.
- Enables arithmetic operations with data blocks in buffers.
- Enables writing serial numbers, MAC addresses and other device-specific parameters into user-selectable shadow areas of target devices.
- Allows writing of user-defined signatures and data blocks into target devices.
- Offers several algorithms for calculating checksums.
- Special DLL for user-defined checksum calculation.
- Writes programming session logs with real time stamps.
- The GUI has a special editor for easy setting of device and algorithm parameters, such as fuses, lock bits, boot loader vectors, etc.
- Comprehensive self-test procedure.

Managing Projects and Configurations
- The software supports unlimited number of projects.
- Project files are protected against hackers and corruption.
- The software ensures data integrity - every data transfer to/from a PC or ATE system or memory card is accompanied with CRC sum.
- The software allows storing and retrieving the state of user interface: configurations, colors, fonts, hot keys and other settable preferences.
- Battery powered option allows storing 4 projects on internal memory card; user-selectable by pressing the button on battery compartment.

Computer Control Methods
- From Automated Test Equipment (ATE), In-Circuit Test System (ICT), or programming fixtures.
- From command line or via Application Control Interface (DLL).
- Integration with National Instruments® LabVIEW™ software.
- On-the-fly management utility allows control of already launched and running device programmer.
- Built-in scripting language for writing user scripts. Auto programming can be started by closing fixture lid or by connecting a device.
- Friendly and intuitive graphical user interface (GUI) for creating and debugging projects.
- Optional simplified user interface for unskilled personnel.

Standalone Control
- The programmer can work in a standalone mode that does not require connection to a computer.
- Up to 255 standalone projects can be stored on a built-in memory card.

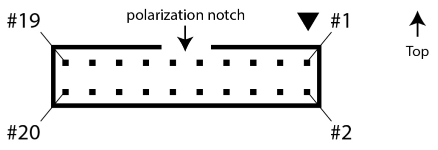
- Any project can be launched by ATE signals or from a computer.
- Special utility allows monitoring standalone activity on a computer.

1.2.4 Connector TARGET

TARGET connector

The **TARGET** connector positioned on the front panel enables connecting a CPI2-B1 device programmer to the target device by the 20-wire ribbon cable included in a CPI2-B1 kit. See here the connector pin assignment and description of the signals in the matrix below.

CPI2-B1 TARGET connector



Pin#	Signal	Signal description – all signals are bidirectional
1	P1	Log 0/1, Vcc or GND
2	P11	Log 0/1, Vcc, Vpp or GND
3	P2	Log 0/1, Vcc or GND
4	GND	Ground
5	P3	Log 0/1, Vcc or GND
6	GND	Ground
7	P4	Log 0/1, Vcc or GND
8	GND	Ground
9	P5	Log 0/1, Vcc or GND
10	GND	Ground
11	P6	Log 0/1, Vcc or GND
12	GND	Ground
13	P7	Log 0/1, Vcc or GND
14	GND	Ground
15	P8	Log 0/1, Vcc or GND
16	GND	Ground
17	P9	Log 0/1, Vcc or GND
18	GND	Ground
19	P10	Log 0/1, Vcc or GND
20	P12	Log 0/1, Vcc, Vpp or GND

- P1 to P10 - logical signals formed by high-speed buffers that can output target-specific logic 0 or 1, Vcc or GND levels, according to the chosen target device type. These lines can output Vcc with levels from 1.2 to 5.5V @ up to 350mA. The buffers are bidirectional, also serving as inputs when the CPI2-B1 programmer reads data.

- P11, P12 – signals formed by high speed mixed-signal circuits that can also output target-specific logic 0 or 1, Vcc or GND levels according to the type of the chosen target device. These lines can output Vcc with levels from 1.2 to 5.5V @ up to 350mA. The mixed-signal buffers are bidirectional, also serving as inputs when the CPI2-B1 programmer reads data. In addition, these two signals can output Vpp voltage with levels from 1.5V to 15V @ up to 100mA.

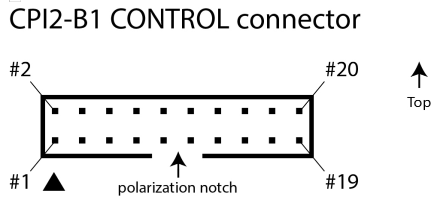
The P1…P12 signals are target-specific. A CPI2-B1 user must ensure that the target device (DUT) is properly connected, according to the target-specific wiring diagram published on the http://phyton.com/products/isp/chipprog-isp2-family/cpi2-b1-connecting web page. When programmer is controlled by the GUI, the same diagram can be viewed in a browser by clicking the **Connection to the target device** link in the **Device Information** window.

To "cut off" the target in the stand-by mode or after completion of any programming operation, CPI2-B1 programmer leaves the P1…P12 signals in high impedance state.

1.2.5 Connector CONTROL

CONTROL connector

The **CONTROL** connector positioned on the right side of the CPI2-B1 unit enables connecting the programmer to Automated Test Equipment (ATE) or the fixture by the 20-wire ribbon cable. See here the connector pin assignment and description of the signals in the matrix below. Since the programmer can be optionally equipped with a CPI2-ISO IO galvanic isolation board with RS232 interface, there are two different diagrams shown below.



Variant WITHOUT optical isolation (CPI2-ISO is NOT installed inside of CPI2-B1)			
Pin#	**Signal**	**Type of signal**	**Signal description – all signals are bidirectional**
1	GND	Ground	Ground
2	GND	Ground	Ground
3	PROJ_SEL0	< Input	Project select 0; active log 1
4	START	< Input	Control signal that launches/stops programming; active: log 0
5	PROJ_SEL1	< Input	Project select 1; active: log 1
6	5V_CHARGE	Output >	5V @ 500 mA sending to battery compartment for charging the

			battery
7	PROJ_SEL2	< Input	Project select 2; active: log 1
8	5V_IN	< Input	5V input - either from external power supply or the CPI2-B1 battery
9	PROJ_SEL3	< Input	Project select 3; active: log 1
10	5V_IN	< Input	5V input - either from external power supply or the CPI2-B1 battery
11	PROJ_SEL4	< Input	Project select 4; active: log 1
12	GND	Ground	Ground
13	SAMODE	< Input	Standalone mode control; active: log 1
14	GND	Ground	Ground
15	ST_GOOD	Output >	Signal GOOD sent to ATE; active: log 0
16	GND	Ground	Ground
17	ST_BUSY	Output >	Signal BUSY sent to ATE; active: log 0
18	NC	Not connected	Not connected
19	ST_ERROR	Output >	Signal ERROR sent to ATE; active: log 0
20	NC	Not connected	Not connected

- **PROJ_SEL[4..0]** – 5-bit selector for choosing one of 32 preloaded projects - the #0 project select code is 000000, the #4 project - 000100;
- **ST_GOOD | ST_ERROR | ST_BUSY** - programmer status lines; active status: log 0;
- **START** - External signal launching and stopping the programmer; active status: log 0. If this signal remains applied to this connector pin for longer than 2 sec it switches the programmer to the Standalone Mode;
- **5V_CHARGE -** +5V @ 500mA max signal that charges CPI2-BB battery. It can be used for powering on the project selector;
- **5V_IN** – 5V supplied either from an external power adapter plugged to the programmer or from a stacked CPI2-BB compartment with a built-in battery or floating 0V if both external power adapter and CPI2-BB compartment are not connected to the CPI2-B1 unit.
- **SAMODE** - Standalone mode control; log 1 applied to this input at a moment of powering the programmer on switches the programmer to the standalone mode.

Variant WITH optical isolation (CPI2-ISO is installed inside of CPI2-B1)			
Pin#	**Signal**	**Type of signal**	**Signal description – all signals are bidirectional**
1	NC	Not connected	Not connected
2	NC	Not connected	Not connected
3	PROJ_SEL0	< Input	Optically isolated project select 0; active log 1
4	START	< Input	Optically isolated control signal that launches/stops programming; active: log 0

5	PROJ_SEL1	< Input	Optically isolated project select 1; active: log 1
6	V_ISO	Output >	Optically isolated 5V @ 10 mA max
7	PROJ_SEL2	< Input	Optically isolated project select 2; active: log 1
8	NC	Not connected	Not connected
9	PROJ_SEL3	< Input	Optically isolated project select 3; active: log 1
10	NC	Not connected	Not connected
11	PROJ_SEL4	< Input	Optically isolated project select 4; active: log 1
12	GND	Ground	Optically isolated GND line
13	SAMODE	< Input	Standalone mode control; active: log 1
14	GND_ISO	Ground	Optically isolated GND line
15	ST_GOOD	Output >	Optically isolated signal GOOD sent to ATE; active: log 0
16	GND_ISO	Ground	Optically isolated GND line
17	ST_BUSY	Output >	Optically isolated signal BUSY sent to ATE; active: log 0
18	RS232_TX	Output >	Data transmitted to computer
19	ST_ERROR	Output >	Optically isolated signal ERROR sent to ATE; active: log 0
20	RS232_RX	< Input	Not connected

- **PROJ_SEL[4..0]** – 5-bit selector for choosing one of 32 preloaded projects - the #0 project select code is 000000, the #4 project - 000100;
- **ST_GOOD | ST_ERROR | ST_BUSY** - Optically isolated programmer status lines; active status: log 0;
- **START** - Optically isolated external signal launching and stopping the programmer; active status: log 0; If this signal remains applied to this connector pin for longer than 2 sec it switches the programmer to the Standalone Mode;
- **5V_CHARGE** +5V @ 500mA max signal that charges CPI2-BB battery. It can be used as a power source for the project selector;
- **5V_IN** – 5V supplied either from an external power adapter plugged to the programmer or from a stacked CPI2-BB compartment with a built-in battery or floating 0V if both external power adapter and CPI2-BB compartment are not connected to the CPI2-B1 unit.
- **SAMODE** - Standalone mode control; log 1 applied to this input at a moment of powering the programmer on switches the programmer to the standalone mode.

1.2.6 Single- and Gang-programming control modes

ChipProg-02 software allows to drive CPI2-B1 device programmers in two different modes:

- **Single-programming** mode for programming one target device at a time by means of one CPI2-B1 programmer.

- **Gang-programming** mode for simultaneous programming of multiple devices by means of multiple CPI2-B1 programmers driven from one PC. This mode is intended for mass production in test fixtures or other ATE.

The programming mode is set in the **Startup** dialog by checking and unchecking the **Gang Mode** checkbox.