Contents

About This Manual xv
Customer Support xix

Part I Getting Started

Chapter 1 What Is C*? 3

1.1 C* and C 3
1.2 C* Implementations 4
1.3 Program Development Facilities 4

Chapter 2 Using C* 7

2.1 Step 1: Declaring Shapes and Parallel Variables 8
2.1.1 Shapes 8
2.1.2 Parallel Variables 9
2.1.3 Scalar Variables 10
2.2 Step 2: Selecting a Shape 11
2.3 Step 3: Assigning Values to Parallel Variables 11
2.4 Step 4: Performing Computations Using Parallel Variables 12
2.5 Step 5: Choosing an Individual Element of a Parallel Variable 13
2.6 Step 6: Performing a Reduction Assignment of a Parallel Variable 14
2.7 Compiling and Executing the Program 15
2.7.1 Compiling 15
2.7.2 Executing 16

Copyright © 1990–1995 Thinking Machines Corporation

Part II Programming in C*

Chapter 3 Using Shapes and Parallel Variables 19

3.1 What Is a Shape? 19

3.2 Choosing a Shape 20

3.3 Declaring a Shape 21
3.3.1 Declaring More Than One Shape 22
3.3.2 The Scope of a Shape 22

3.4 Obtaining Information about a Shape 22

3.5 More about Shapes 23

3.6 What Is a Parallel Variable? 24
3.6.1 Parallel and Scalar Variables 24

3.7 Declaring a Parallel Variable 25
3.7.1 Declaring More Than One Parallel Variable 26
A Shortcut for Declaring More Than One Parallel Variable 26
3.7.2 Positions and Elements 27
3.7.3 The Scope of Parallel Variables 28

3.8 Declaring a Parallel Structure 28

3.9 Declaring a Parallel Array 30

3.10 Initializing Parallel Variables 32
3.10.1 Initializing Parallel Structures and Parallel Arrays 32

3.11 Obtaining Information about Parallel Variables 32
3.11.1 The **`positionsof`**, **`rankof`**, and **`dimof`** Intrinsic Functions . . 33
3.11.2 The **`shapeof`** Intrinsic Function 33

3.12 Choosing an Individual Element of a Parallel Variable 34
3.12.1 Precedence 35

Chapter 4 Choosing a Shape 37

4.1 The **`with`** Statement 37
4.1.1 Default Shape 39
4.1.2 Using a Shape-Valued Expression 39

4.2 Nesting **`with`** Statements 39

4.3 Initializing a Variable at Block Scope 41

4.4 Parallel Variables Not of the Current Shape 42

Version 7.2, May 1995
Copyright © 1990–1995 Thinking Machines Corporation

Chapter 5 Using C* Operators and Data Types 43

5.1 Standard C Operators 43
5.1.1 With Scalar Operands 43
5.1.2 With a Scalar Operand and a Parallel Operand 44
Assignment with a Parallel LHS and a Scalar RHS 44
Assignment with a Scalar LHS and a Parallel RHS 45
5.1.3 With Two Parallel Operands 47
5.1.4 Unary Operators for Parallel Variables 48
5.1.5 The Conditional Expression 49

5.2 New C* Operators 50
5.2.1 The `<?` and `>?` Operators 50
5.2.2 The `%%` Operator 51

5.3 Reduction Operators 52
5.3.1 Unary Reduction 53
5.3.2 Parallel-to-Parallel Reduction Assignment 54
5.3.3 List of Reduction Operators 54
5.3.4 The `-=` Reduction Operator 55
5.3.5 The `*=` and `/=` Reduction Operators 56
5.3.6 Minimum and Maximum Reduction Operators 56
5.3.7 Bitwise Reduction Operators 56
Bitwise OR 57
Bitwise AND 57
Bitwise Exclusive OR 57
5.3.8 Reduction Assignment Operators with a Parallel LHS 58
5.3.9 Precedence of Reduction Operators 58

5.4 The `bool` Data Type 58
5.4.1 The `boolsizeof` Operator 59
With a Parallel Variable or Data Type 59
With a Scalar Variable or Data Type 60

5.5 64-Bit Integers 60
5.5.1 Declaring 64-Bit Integers 60
5.5.2 Integer Constants 61
5.5.3 The Semantics of 64-Bit Integers 61
5.5.4 Hints on Using 64-Bit Integers 61
5.5.5 Performance 62
5.5.6 Library Functions 63
Communication Functions 63
Standard C Functions 63
Mathematical Functions 64
String Functions 64

Copyright © 1990–1995 Thinking Machines Corporation

I/O Functions 64
5.6 Parallel Unions 65
5.6.1 Limitations 65
5.7 Parallel Enumeration Type 66

Chapter 6 Setting the Context 67

6.1 The **where** Statement 67
6.1.1 The **else** Clause 69
6.1.2 The **where** Statement and **positionsof** 70
6.1.3 The **where** Statement and Parallel-to-Scalar Assignment 71
6.2 The **where** Statement and Scalar Code 71
6.3 Nesting **where** and **with** Statements 72
6.3.1 Nesting **where** Statements 72
6.3.2 Nesting **with** Statements 73
6.3.3 The **break**, **goto**, **continue**, and **return** Statements 74
6.4 The **everywhere** Statement 75
6.5 When There Are No Active Positions 76
6.5.1 When There Is a Reduction Assignment Operator 77
Unary Reduction Operators 77
Binary Reduction Assignment Operators 78
6.5.2 Preventing Code from Executing 78
6.6 Looping through All Positions 79
6.7 Context and the **||**, **&&**, and **?:** Operators 81
6.7.1 **||** and **&&** 81
6.7.2 The **?:** Operator 84

Chapter 7 Pointers 85

7.1 Scalar-to-Scalar Pointers 85
7.2 Scalar Pointers to Shapes 86
7.3 Scalar Pointers to Parallel Variables 86
7.3.1 Alternative Declaration Syntax Not Allowed 88
7.3.2 Arrays 89
7.3.3 Pointer Arithmetic 89
7.3.4 Parallel Indexes into Parallel Arrays 90
Adding a Parallel Variable to a Pointer to a Parallel Variable 92
Limitations 93

Version 7.2, May 1995
Copyright © 1990–1995 Thinking Machines Corporation

Chapter 8 Functions **95**

8.1 Using Parallel Variables with Functions 95
8.1.1 Passing a Parallel Variable as an Argument 95
If the Parallel Variable Is Not of the Current Shape 96
8.1.2 Returning a Parallel Variable 97
In a Nested Context 97

8.2 Passing by Value and Passing by Reference 98

8.3 Using Shapes with Functions 100
8.3.1 Passing a Shape as an Argument 100
8.3.2 Returning a Shape 101

8.4 When You Don't Know What the Shape Will Be 101
8.4.1 The **`current`** Predeclared Shape Name 101
8.4.2 The **`void`** Predeclared Shape Name 102
Using **`shapeof`** with the **`void`** Shape 103
Using **`void`** when Returning a Pointer 104

8.5 Overloading Functions 104

Chapter 9 More on Shapes and Parallel Variables **107**

9.1 Partially Specifying a Shape 107
9.1.1 Partially Specifying an Array of Shapes 108
Arrays and Pointers 109
9.1.2 Limitations 109

9.2 Creating Copies of Shapes 110
9.2.1 Assigning a Local Shape to a Global Shape 111

9.3 Dynamically Allocating a Shape 112

9.4 Deallocating a Shape 113

9.5 Dynamically Allocating a Parallel Variable 114

9.6 Casting with Shapes and Parallel Variables 116
9.6.1 Scalar-to-Parallel Casts 116
9.6.2 Parallel-to-Parallel Casts 116
Casts to a Different Type 117
Casts to a Different Shape 117
9.6.3 With a Shape-Valued Expression 118
9.6.4 Parallel-to-Scalar Casts 118

9.7 Declaring a Parallel Variable with a Shape-Valued Expression 118

9.8 The **`physical`** Shape 119

Copyright © 1990–1995 Thinking Machines Corporation

Chapter 10 Communication 121

10.1 Using a Parallel Left Index for a Parallel Variable 122
10.1.1 A Get Operation 123
10.1.2 A Send Operation 124
10.1.3 Use of the Index Variable 125
10.1.4 If the Shape Has More Than One Dimension 126
10.1.5 When There Are Potential Collisions 127
For a Get Operation 127
For a Send Operation 128
10.1.6 When There Are Inactive Positions 130
For a Get Operation 130
For a Send Operation 131
Send and Get Operations in Function Calls 133
10.1.7 Mapping a Parallel Variable to Another Shape 134
10.1.8 Limitation of Using Parallel Variables with a Parallel Left Index 136
10.1.9 What Can Be Left-Indexed 136
10.1.10 An Example: Adding Diagonals in a Matrix 137

10.2 Using the **`pcoord`** Function 139
10.2.1 An Example 142

10.3 The **`pcoord`** Function and Grid Communication 143
10.3.1 Grid Communication without Wrapping 144
10.3.2 Grid Communication with Wrapping 145

Part III C* Communication Functions

Chapter 11 Introduction to the C* Communication Library 149

11.1 Two Kinds of Communication 150
11.1.1 Grid Communication 150
11.1.2 General Communication 151

11.2 Communication and Computation 151

Chapter 12 Grid Communication 153

12.1 Aspects of Grid Communication 153
12.1.1 Axis 154
12.1.2 Direction 154
12.1.3 Distance 155

Version 7.2, May 1995
Copyright © 1990–1995 Thinking Machines Corporation

12.1.4 Border Behavior 155
12.1.5 Behavior of Inactive Positions 155
12.2 The **from_grid_dim** Function 156
12.2.1 With Arithmetic Types 157
Examples 158
When Positions Are Inactive 161
12.2.2 With Parallel Data of Any Length 162
12.3 The **from_grid** Function 163
12.3.1 With Arithmetic Types 163
12.3.2 With Parallel Data of Any Length 165
12.4 The **to_grid** and **to_grid_dim** Functions 165
12.4.1 With Arithmetic Types 166
When Positions Are Inactive 167
Examples 167
12.4.2 With Parallel Data of Any Length 169
12.5 The **from_torus** and **from_torus_dim** Functions 170
12.5.1 With Arithmetic Types 170
12.5.2 With Parallel Data of Any Length 173
12.6 The **to_torus** and **to_torus_dim** Functions 173
12.6.1 With Arithmetic Types 173
Examples 175
12.6.2 With Parallel Data of Any Length 177

Chapter 13 Communication with Computation 179

13.1 What Kinds of Computation? 179
13.2 Choosing Elements 180
13.2.1 The Scan Class 180
The Scan Subclass 183
13.2.2 The Scan Set 183
Inclusive and Exclusive Operations 185
13.2.3 Segment Bits and Start Bits 186
If smode Is **CMC_segment_bit** 186
If smode Is **CMC_start_bit** 186
Inactive Positions 187
The Direction of the Operation 188
Data from Another Scan Set 189
13.3 The **scan** Function 190
13.3.1 With Arithmetic Types 190
13.3.2 Examples 192

Copyright © 1990–1995 Thinking Machines Corporation

13.3.3 With Parallel Data of Any Length 194
13.4 The **reduce** and **copy_reduce** Functions 195
13.4.1 The **reduce** Function 195
An Example 197
13.4.2 The **copy_reduce** Function 197
An Example 198
13.5 The **spread** Function 199
An Example 200
13.6 The **copy_spread** Function 200
13.6.1 With Arithmetic Types 201
An Example 201
13.6.2 With Parallel Data of Any Length 202
13.7 The **enumerate** Function 203
13.7.1 Examples 203
13.8 The **rank** Function 205
13.8.1 Examples 206
13.9 The **multispread** Function 208
13.9.1 The **copy_multispread** Function 212
13.10 The **global** Function 212

Chapter 14 General Communication 215

14.1 The **make_send_address** Function 215
14.1.1 Obtaining a Single Send Address 216
An Example 217
14.1.2 Obtaining Multiple Send Addresses 217
When Positions Are Inactive 218
An Example 218
14.2 Getting Parallel Data: The **get** Function 219
14.2.1 Getting Parallel Variables 220
Collisions in Get Operations 221
14.2.2 Getting Parallel Data of Any Length 222
14.3 Sending Parallel Data: The **send** Function 224
14.3.1 Sending Parallel Variables 224
Inactive Positions 226
An Example 226
14.3.2 Sending Parallel Data of Any Length 228
14.3.3 Sorting Elements by Their Ranks 230
14.4 Communicating between Scalar and Parallel Variables 232
14.4.1 From a Parallel Variable to a Scalar Variable 232

Version 7.2, May 1995
Copyright © 1990–1995 Thinking Machines Corporation

The **read_from_position** Function 232
The **read_from_pvar** Function 234
14.4.2 From a Scalar Variable to a Parallel Variable 236
The **write_to_position** Function 236
The **write_to_pvar** Function 238
14.5 The **make_multi_coord** and **copy_multispread** Functions 240
14.5.1 An Example 243

Appendix A Using allocate_detailed_shape 245

A.1 The Default Layout 246
A.1.1 Physical Grids 246
A.1.2 Garbage Positions 246
A.1.3 Subgrids 248
A.1.4 Axis Sequence 249
A.1.5 Subgrid Sequence 250
A.1.6 Putting It All Together 251
A.2 Layout without Vector Units 252
A.3 Controlling Subgrid Layout: Using Serial Axes and Weighting Axes . . . 253
A.3.1 Serial Axes 254
A.3.2 Weighting Axes 255
A.4 Performance Issues 256
A.4.1 Effect of Subgrid Length and Physical Grid 256
A.4.2 Effect of Serial Axes 257
A.4.3 Effect of Garbage Positions 258
A.5 Determining a Shape's Layout 258
A.6 Using **allocate_detailed_shape** 259
A.6.1 In More Detail 262
A.6.2 Example 265

Appendix B Memory Layout on the CM-5 269

B.1 Memory Layout of Parallel Variables 269
B.2 Pointers to Parallel Variables 271
B.3 Manipulating Pointers to Parallel Variables 272
B.4 Shape Aliasing 273
B.4.1 Examples 273

Copyright © 1990–1995 Thinking Machines Corporation

Appendix C CM-5 C* Table Lookup Utility 279

C.1 An Example 280

Appendix D Global/Local Programming 283

D.1 Overview 284
D.1.1 Limitations 286

D.2 Flow Control in a Global/Local Program 286

D.3 Shape and Context 287

D.4 Global and Local Data 288
D.4.1 Parallel Variables 288
D.4.2 Scalar Variables 289

D.5 I/O 290

D.6 Writing a Global/Local Program 291
D.6.1 Calling the Local Function 292
D.6.2 Writing the Local Function 292
D.6.3 Writing the Prototype File 293
Prototypes 293
Use 294

D.7 Global/Local Inquiry Functions 294
D.7.1 Determining the Global Coordinates from the Local Coordinates 294
Example 295
D.7.2 Determining the Local Coordinates from the Global Coordinates 297
Example 297
D.7.3 Determining the Node Number from the Global Coordinates 298
Example 298

D.8 Compiling and Linking 299

D.9 Debugging 300
D.9.1 Performance Analysis 301

D.10 Sample Program 302
D.10.1 Global Program 302
D.10.2 Local Program 304
D.10.3 Prototype File 305

Appendix E Integer Inquiry Functions 307

E.1 The `leadz` and `leadz_nz` Functions 308

Version 7.2, May 1995
Copyright © 1990–1995 Thinking Machines Corporation

E.1.1 Examples 309
E.2 The **dimulh** and **dumulh** Functions 309
E.3 The **popcnt** Function 310
E.3.1 Examples 311
E.4 The **poppar** Function 311
E.4.1 Examples 312

Appendix F Glossary 313

Index 321

Copyright © 1990–1995 Thinking Machines Corporation