EPICS

Record Reference Manual

Philip Stanley

Janet Anderson

Marty Kraimer

EPICS Release 3.13

Copyright

Initial development by:

> The Controls and Automation Group (AOT-8),
> Ground Test Accelerator,
> Accelerator Technology Division,
> Los Alamos National Laboratory.

Co-developed with:

> The Controls and Computing Group,
> Accelerator Systems Division,
> Advanced Photon Source,
> Argonne National Laboratory.

Contents

Preface. 1

Organization.....1
Conventions.....1
Record Tables.....2
Inaccuracies.....2

Chapter 1: Database Concepts 3

1. Scanning Specification.....3
   Periodic Scanning 4
   Event Scanning 5
   Passive Scanning. 5
   Phase 10
   Forward Process Links 11
2. Address Specification.....12
   Hardware Addresses 12
   Database Addresses 14
   Constants. 15
3. Conversion Specification.....16
   Discrete Conversions 16
   Analog Conversions 17
4. Alarm Specification.....22
5. Monitor Specification.....25
   Notification 25
   List Maintenance. 25
6. Control Specification.....26
   Closing an Analog Control Loop 26

Chapter 2: Fields Common to All Record Types. 28

Introduction.....28
2. Scan Fields.....28
   Field Summary 28
   Field Description. 29
3. Alarm Fields.....30
   Field Summary 30
   Field Description. 31
4. Device Fields.....31
   Field Summary 31
   Field Description. 32
5. Debugging Fields.....32
   Field Summary 32
   Field Description. 32
6. Miscellaneous Fields.....33
   Field Description. 33
   Field Description. 33

Chapter 3: Fields Common to Many Record Types 35

Introduction 35
2. Input Records 35
Common Fields 35
Device Input 36
Soft Input 36
Simulation Mode 37
3. Output Records 38
Common Fields 38
Soft Output 39
Output Mode Select 39
Simulation Mode 40
Invalid Alarm Output Action 40

Chapter 4: ai—Analog Input 41
Introduction 41
2. Scanning Parameters 41
3. Read and Convert Parameters 42
Input Specification 42
Conversion Related Fields 43
4. Operator Display Parameters 44
5. Alarm Parameters 44
6. Monitor Parameters 45
7. Run-Time Parameters and Simulation Mode Parameters 45
8. Record Support Routines 46
9. Record Processing 47
10. Device Support 48
Fields Of Interest To Device Support 48
Device Support Routines 48
Device Support For Soft Records 49

Chapter 5: ao - Analog Output 51
Introduction 51
2. Scan Parameters 51
3. Desired Output Parameters 51
4. Convert and Write Parameters 53
Conversion Related Fields and the Conversion Process 53
Output Specification 54
5. Operator Display Parameters 54
6. Alarm Parameters 55
7. Monitor Parameters 56
8. Run-Time Parameters and Simulation Mode Parameters 56
9. Record Support Routines 57
10. Record Processing 58
11. Device Support 59
Fields Of Interest To Device Support 59
Device Support routines 60
Device Support For Soft Records 61

Chapter 6: The Archive Record 62

Introduction 62
2. Scan Parameters 63
3. Read Parameters 63
4. Archive and Monitor Parameters 63
5. Operator Display Parameters 65
6. Run-time Parameters 65

Chapter 7: bi - Binary Input 67
Introduction 67
2. Scan Parameters 67
3. Read and Convert Parameters 67
Conversion Fields 68
4. Operator Display Parameters 68
5. Alarm Parameters 69
6. Run-time Parameters and Simulation Mode Parameters 69
7. Record Support Routines 70
8. Record Processing 71
9. Device Support 72
Fields Of Interest To Device Support 72
Device Support routines 72
Device Support for Soft Records 73

Chapter 8: bo—Binary Output 74
Introduction 74
2. Scan Parameters 74
3. Desired Output Parameters 74
4. Convert and Write Parameters 75
Conversion Parameters 76
Output Specification 76
5. Operator Display Parameters 76
6. Alarm Parameters 77
7. Run-Time and Simulation Mode Parameters 77
8. Record Support Routines 78
9. Record Processing 79
10. Device Support 80
Fields Of Interest To Device Support 80
Device Support routines 81
Device Support For Soft Records 81

Chapter 9: Calc - Calculation 83
Introduction 83
2. Scan Parameters 83
3. Read Parameters 83
4. Expression 84
Operands 85
Algebraic Operators 85
Trigonometric Operators. 86
Relational Operators 86
Logical Operators 86

Bitwise Operators. 86
Parentheses and Comma 87
Conditional Expression 87
Examples 87
5. Operator Display Parameters 88
6. Alarm Parameters..... 88
7. Monitor Parameters..... 89
8. Run-time Parameters..... 89
9. Record Support Routines 90
10. Record Processing..... 91

Chapter 10: Calcout - Calculation Output Record 93
Introduction 93
2. Scan Parameters 93
3. Read Parameters..... 94
4. Expression..... 94
Operands 95
Algebraic Operators 96
Trigonometric Operators 96
Relational Operators 96
Logical Operators. 97
Bitwise Operators. 97
Parentheses and Comma 97
Conditional Expression 97
Examples 97
5. Output Parameters..... 98
6. Operator Display Parameters 99
7. Alarm Parameters..... 101
8. Monitor Parameters..... 101
9. Run-time Parameters..... 102
10. Record Support Routines 102
11. Record Processing..... 104
process() 104
execOutput(). 104

Chapter 11: compress - Compression 105
Introduction 105
2. Scanning Parameters 105
3. Read Parameters and Algorithm Parameters 105
Input Specification 106
Algorithms and Related Fields. 106
4. Operator Display Parameters 107
5. Alarm Parameters..... 108
6. Run-time Parameters..... 108
7. Record Support Routines 109
8. Record Processing..... 110

Chapter 12: CPID Control. 111
Introduction 111

2. Scan Parameters 111
3. Controlled Variable and Setpoint Parameters 112
4. Expression Parameters 112
5. Output, Readback, and Mode Parameters 114
6. Operator Display Parameters 115
7. Alarm Parameters 116
8. Monitor Parameters 117
9. Run-time Parameters 117
10. Record Support Routines 118
11. Record Processing 119
12. Device Support 120

Chapter 13: dfanout 121

Introduction 121
2. Scan Parameters 121
3. Desired Output Parameters 121
4. Write Parameters 122
5. Operator Display Parameters 122
6. Alarm Parameters 123
7. Monitor Parameters 124
8. Run-Time Parameters and Simulation Mode Parameters 124
9. Record Support Routines 124
10. Record Processing 125

Chapter 14: Event 126

Introduction 126
2. Scan Parameters 126
3. Input Specification 126
4. Event Number Parameters 127
5. Operator Display Parameters 127
6. Alarm Parameters 127
7. Simulation Mode Parameters 128
8. Record Support Routines 128
9. Record Processing 128
10. Device Support 129
Fields of Interest To Device Support 129
Device Support Routines 129
Device Support For Soft Records 130

Chapter 15: Fanout 131

Introduction 131
2. Scan Parameters 131
3. Operator Display Parameters 132
4. Alarm Parameters 132
5. Run-time Parameters 133
6. Record Support Routines 133
7. Record Processing 133

Chapter 16: Histogram . . . 134
Introduction ... 134
2. Scanning Parameters ... 134
3. Read Parameters ... 134
4. Operator Display Parameters ... 135
5. Alarm Parameters ... 135
6. Monitor Parameters ... 135
7. Run-time and Simulation Mode Parameters ... 136
8. Record Support Routines ... 137
9. Record Processing ... 138
10. Device Support ... 139
Fields Of Interest To Device Support . . . 139
Device Support Routines . . . 139
Device Support For Soft Records . . . 139

Chapter 17: longin—Long Input . . . 141
Introduction ... 141
2. Scan Parameters ... 141
3. Read Parameters ... 141
4. Operator Display Parameters ... 142
5. Alarm Parameters ... 142
6. Monitor Parameters ... 143
7. Run-time and Simulation Mode Parameters ... 144
8. Record Support Routines ... 144
9. Record Processing ... 145
10. Device Support ... 146
Fields Of Interest To Device Support . . . 146
Device Support routines . . . 146
Device Support For Soft Records . . . 147

Chapter 18: longout - Long Output . . . 148
Introduction ... 148
2. Scan Parameters ... 148
3. Desired Output Parameters ... 148
4. Write Parameters ... 149
5. Operator Display Parameters ... 149
6. Alarm Parameters ... 150
7. Monitor Parameters ... 151
8. Run-time and Simulation Mode Parameters ... 151
9. Record Support Routines ... 152
10. Record Processing ... 153
11. Device Support ... 154
Fields Of Interest To Device Support . . . 154
Device Support Routines . . . 154
Device Support For Soft Records . . . 155

Chapter 19: mbbi — Multi-Bit Binary Input . . . 156
Introduction ... 156

2. Scan parameters 156
3. Read and Convert Parameters 157
4. Operator Display Parameters 159
5. Alarm Parameters 159
6. Run-time and Simulation Mode Parameters 160
7. Record Support Routines 161
8. Record Processing 162
9. Device Support 163
Fields Of Interest To Device Support 163
Device Support Routines 164
Device Support For Soft Records 164

Chapter 20: mbbiDirect - Multi-Bit Binary Input Direct 166
Introduction 166
2. Scan fields 166
3. Read and Convert fields 166
4. Operator Display Parameters 168
5. Alarm Parameters 168
6. Run-time and Simulation Mode Fields 168
7. Record Support Routines 169
8. Record Processing 169
9. Device Support 170
Fields Of Interest To Device Support 170
Device Support Routines 171
Device Support For Soft Records 172

Chapter 21: mbbo — Multi-Bit Binary Output 173
Introduction 173
2. Scan Parameters 173
3. Desired Output Parameters 173
4. Convert and Write Parameters 174
5. Operator Display Parameters 176
6. Alarm Parameters 176
7. Run-Time and Simulation Mode Parameters 177
8. Record Support Routines 178
9. Record Processing 179
10. Device Support 180
Fields Of Interest To Device Support 180
Device Support Routines 181
Device Support For Soft Records 182

Chapter 22: mbboDirect - Multi-Bit Binary Output Direct 183
Introduction 183
2. Scan Parameters 183
3. Desired Output Parameters 183
4. Convert and Write Parameters 184
5. Operator Display Parameters 185
6. Alarm Parameters 185

7. Run-time and Simulation Mode Parameters 186
8. Record Support Routines 187
9. Record Processing 187
10. Device Support 188
Fields Of Interest To Device Support 188
Device Support Routines 189
Device Support For Soft Records 190

Chapter 23: Permissive. 191
Introduction 191
2. Scan Parameters 191
3. Client-server Parameters 191
4. Operator Display Parameters 192
5. Alarm Parameters 192
6. Run-time Parameters 192
7. Record Support Routines 193

Chapter 24: PID Control 194
Introduction 194
2. Scan Parameters 194
3. Controlled Variable Parameters 195
4. Setpoint Parameters 195
5. Expression Parameters 196
6. Operator Display Parameters 197
7. Alarm Parameters 197
8. Monitor Parameters 198
9. Run-time Parameters 198
10. Record Support Routines 199
11. Record Processing 200

Chapter 25: pulseCounter 202
Introduction 202
2. Scan Parameters 202
3. Setup Parameters 202
4. Write Parameters 203
5. Operator Display Parameters 204
6. Alarm Parameters 204
7. Run-time parameters 204
8. Record Support Routines 205
9. Record Processing 206
10. Device Support 206
Fields Of Interest To Device Support 206
Device Support Routines 207

Chapter 26: pulseDelay 208
Introduction 208
2. Scan Parameters 208
3. Trigger Parameters 208

4. Pulse Parameters 209
5. Operator Display Parameters 210
6. Alarm Parameters....210
7. Run-time Parameters....211
8. Record Support Routines 211
9. Record Processing....212
10. Device Support....213
Fields Of Interest To Device Support 213
Device Support Routines 214

Chapter 27: pulseTrain 215
Introduction 215
2. Scan Parameters....215
3. Trigger Parameters....215
4. Pulse Parameters 216
5. Output Parameters....216
6. Operator Display Parameters 217
7. Alarm Parameters....217
8. Run-time Parameters....217
9. Record Support Routines 218
10. Record Processing....219
11. Device Support....219
Fields Of Interest To Device Support 219
Device Support Routines 220
Soft Device Support 221

Chapter 28: scan 222
1. Introduction....222
A Simple Single Dimensional Scan 222
Two Dimensional Scanning 223
2. Scan Parameters....224
Positioner Parameters 224
Linear Mode 225
Position Verification, Readback Process Variable, and Delta Parameters 228
Detector Trigger Process Variables and Desired Command 228
3. Data Acquisition Parameters....229
4. Operator Display Parameters 230
5. Run-time Parameters....231

Chapter 29: sel - Select 235
Introduction 235
2. Scan Parameters....235
3. Read Parameters....235
4. Select Parameters 237
5. Operator Display Parameters 237
6. Alarm Parameters....238
7. Monitor Parameters....239
8. Run-time Parameters....239

9. Record Support Routines 240
10. Record Processing 241

Chapter 30: seq - Sequence 242
Introduction 242
2. Scan Parameters 242
3. Desired Output Parameters 242
4. Output Parameters 244
5. Selection Algorithm Parameters 244
6. Delay Parameters 245
7. Operator Display Parameters 245
8. Alarm Parameters 246
9. Record Support Routines 246

Chapter 31: State 247
Introduction 247
2. Scan Parameters 247
3. Operator Display Parameters 247
4. Alarm Parameters 248
5. Run-time Parameters 248
6. Record Support Routines 248

Chapter 32: Stepper Motor 249
Introduction 249
2. Scan Parameters 249
3. Setup Parameters 250
4. Desired Output Parameters 251
5. Output and Readback Parameters 251
6. Operator Display Parameters 252
7. Alarm Parameters 252
8. Monitor Parameters 253
9. Run-time Parameters 253
10. Record Support Routines 255
11. Record Processing 256
12. Device Support 256

Chapter 33: stringin - String Input 257
Introduction 257
2. Scan Parameters 257
3. Read Parameters 257
4. Operator Display Parameters 258
5. Alarm Parameters 258
6. Run-time and Simulation Mode Parameters 258
7. Record Support Routines 259
8. Record Processing 259
9. Device Support 260
Fields Of Interest To Device Support 260

Device Support Routines 260
Device Support For Soft Records 261

Chapter 34: stringout — String Output 262
Introduction 262
2. Scan Parameters 262
3. Desired Output Parameters 262
4. Write Parameters..263
5. Operator Display Parameters 263
6. Alarm Parameters...264
7. Run-time and Simulation Mode Parameters 264
8. Record Support Routines 265
9. Record Processing..265
10. Device Support...266
Fields Of Interest To Device Support 266
Device Support Routines 266
Device Support for Soft Records 267

Chapter 35: subArray. 268
Introduction 268
2. Scan Parameters 268
3. Read Parameters..269
4. Array Parameters 269
5. Operator Display Parameters 270
6. Alarm Parameters...270
7. Run-time Parameters...270
8. Record Support Routines 271
9. Record Processing..272
10. Device Support...273
Fields Of Interest To Device Support 273
Device Support Routines 273
Device Support For Soft Records 274

Chapter 36: sub - Subroutine 275
Introduction 275
2. Scan Parameters 275
3. Read Parameters..275
4. Subroutine Connection...277
5. Operator Display Parameters 277
6. Alarm Parameters...278
7. Monitor Parameters...278
8. Run-time Parameters...279
9. Record Support Routines 280
10. Record Processing...281
11. Example Synchronous Subroutine 281
12. Example Asynchronous Subroutine...282

Chapter 37: Timer 284

Introduction 284
2. Scan Parameters 284
3. Setup Parameters 285
4. Write and Convert Parameters 285
5. Operator Display Parameters 286
6. Alarm Parameters 286
7. Event Generation Parameters 287
8. Run-time Parameters 287

Chapter 38: Wait. 289
Introduction 289
2. Scan Parameters 290
3. Read Parameters 290
4. Expression-related Parameters 291
Operands 292
Algebraic Operators 292
Trigonometric Operators 293
Relational Operators 293
Logical Operators. 293
Bitwise Operators. 293
parentheses and Comma. 294
Conditional Expression 294
Examples 294
5. Desired Output Parameters 295
6. Write Parameters 295
7. Operator Display Parameters 296
8. Alarm Parameters 296
9. Monitor Parameters 297
10. Run-time Parameters 297

Chapter 39: Waveform. 299
Introduction 299
2. Scan Parameters 299
3. Read Parameters 299
4. Operator Display Parameters 300
5. Alarm Parameters 300
6. Run-time Parameters 301
7. Record Support Routines 301
8. Record Processing 303
9. Device Support 303
Fields Of Interest To Device Support 303
Device Support Routines 304
Device Support For Soft Records 305

Appendix A: Menu Choices 306
GBLCHOICE and RECCHOICE Fields 306
Determining the Choices of a GBLCHOICE or RECCHOICE Field 307
2. Standard Menu Definitions 309

Preface

This is a new version of the EPICS Record Reference Manual. It replaces the format and organization of the previous versions. It was rewritten to make the EPICS database and its records easier to understand, and an introductory chapter, *Chapter 1*, has been added to explain basic database concepts, which will hopefully be very helpful to new EPICS users. The remaining chapters all discuss each individual record, occasionally referring back to the introductory chapter when basic concepts are referred to, i.e., how monitors work, how database scanning works, etc.

Although this manual has attempted to be as thorough as possible, it is not yet complete. Hopefully, with more work and additions, it will be something the user can refer to in order to answer just about any question on any detail of any particular record.

Organization

This manual consists of an introductory chapter that explains basic database concepts, (*Chapter 1*), a chapter which explains fields common to all records, a third chapter which explains fields common to many records, and chapters which explain each individual record. The chapter on fields common to all records summarizes the characteristics of the fields which all records contain, though all records may not use all of them. The chapters on the individual records discuss the fields particular to each record. Of course, there is some overlap because some of the fields common to all record types function peculiarly within an individual record; however, this overlap has been kept to a minimum.

Conventions

There are many typeface and font variations intended to help the reader. The most important is the use of the Courier font to indicate literal strings, strings which should be entered as they appear in a field. For instance, some fields have a set menu of choices. When configuring the database, the choice which the user enters in the field must exactly match one of the menu choices (at least for those who are use Capfast as their database configuration tool). These choices appear in Courier font so that the reader can distinguish them from surrounding text. For instance, here is a passage from the analog output record explaining the OMSL field:

> The first field that determines where the desired output originates is the output mode select field (OSML), which can have two possible values—`closed_loop` or `supervisory`. If `supervisory` is specified, the value in the VAL field can be set externally via dbPuts at run-time.

Here `supervisory` and `closed_loop`, the two possible choices in the OMSL fields, appear in Courier font and must be entered as is.

Record Tables

The chapter on the fields common to all record types, the chapter on the fields common to many record types, and the chapters on each record contain tables which present certain characteristics about each field. The format of these tables appears similar to the following which comes from the CPID record:

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OUT	Output Link	OUTLINK	Yes	0	No	No	N/A	No
ORBL	Output Readback Location	INLINK	Yes		No	No	No	No
more fields	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
more fields	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

The tables present the following information:

The **Field** column contains the literal name of the particular field as it appears in the database.

The **Summary** column contains the full name of the field.

The **Type** column gives the EPICS type of the field. These types are defined in the **<dbFldTypes.h>**.

The **DCT** column indicates whether the field is configurable prior to run-time.

The **Initial** column contains the default initialization value of the field.

The **Access** column indicates whether the field can be accessed by the database routines for reading purposes.

The **Modify** column indicates whether the field can be changed at run-time, for example, by the operator.

The **Rec Proc Monitor** column indicates whether or not the field is monitored at run-time for changes.

The **PP** or process passive column indicates whether writing to this field will cause the record to process.

Inaccuracies

There may be some inaccuracies when describing each of the records in Chapters 3-37. Each of these chapters describes an individual record. Considering the number of record types and support modules now included in the EPICS release and the rate at which they are updated, it's certain that some of the information contained in the description of each of the records either is outdated or was never accurate. In either case, if an inaccuracy is noted, please bring to the attention of the author by sending a brief message describing where the inaccuracy is and what it is (stanley@luke.atdiv.lanl.gov).

Chapter 1: Database Concepts

Bob Dalesio and Philip Stanley
LANSCE Division Automation and Controls Group (LANSCE-8)
Los Alamos National Laboratory

This chapter covers the general functionality that is found in all database records. The topics covered are I/O scanning, I/O address specification, data conversions, alarms, database monitoring, and continuous control:

Section 1, *Scanning Specification*, describes the various conditions under which a record is processed.

Section 2, *Address Specification*, explains the source of inputs and the destination of outputs.

Section 3, *Conversion Specification*, covers data conversions from transducer interfaces to engineering units.

Section 4, *Alarm Specification*, presents the many alarm detection mechanisms available in the database.

Section 5, *Monitor Specification*, details the mechanism which notifies operators about database value changes.

Section 6, *Control Specification*, explains the features available for achieving continuous control in the database.

These concepts are essential in order to understand how the database interfaces with the process.

1. Scanning Specification

Scanning determines when a record is processed. A record is *processed* when it processes its data and performs any actions related to that data. For example, when an output record is processed, it fetches the value which it is to output, converts the value, and then writes that value to the specified location. Each record must specify the scanning method that determines when it will be processed. There are three scanning methods for database records: (1) periodic, (2) event, and (3) passive.

1. Periodic scanning occurs on set time intervals.
2. Event scanning occurs on either an I/O interrupt event or a user-defined event.
3. Passive scanning occurs when the records linked to the passive record are scanned, or when a value is “put” into a passive record through the database access routines.

For periodic or event scanning, the user can also control the order in which a set of records is processed by using the `PHASE` mechanism. For event scanning, the user can control the priority at which a record will process. In addition to the scan and the phase mechanisms, there are data links and forward processing links that can be used to cause processing in other records. This section explains these concepts.

1.1. Periodic Scanning

The following periods for scanning database records are available, though EPICS can be configured to recognize more scan periods:

```
10 second
 5 second
 2 second
 1 second
.5 second
.2 second
.1 second
```

The period that best fits the nature of the signal should be specified. A five-second interval is adequate for the temperature of a mass of water because it does not change rapidly. However, some power levels may change very rapidly, so they need to be scanned every 0.5 seconds. In the case of a continuous control loop, where the process variable being controlled can change quickly, the 0.1 second interval may be the best choice.

For a record to scan periodically, a valid choice must be entered in its `SCAN` field. Actually, the available choices depend on the configuration of the `menuScan.dbd` file. As with most other fields which consists of a menu of choices, the choices available for the SCAN field can be changed by editing the appropriate `.dbd` (database definition) file. `dbd` files are ASCII files that are used to generate header files that are, in turn, are used to compile the database code. Many `dbd` files can be used to configure other things besides the choices of menu fields.

Here is an example of a `menuScan.dbd` file, which has the default menu choices for all periods listed above as well as choices for event scanning, passive scanning, and I/O interrupt scanning:

```
menu(menuScan) {
      choice(menuScanPassive,"Passive")
      choice(menuScanEvent,"Event")
      choice(menuScanI_O_Intr,"I/O Intr")
      choice(menuScan10_second,"10 second")
      choice(menuScan5_second,"5 second")
      choice(menuScan2_second,"2 second")
      choice(menuScan1_second,"1 second")
      choice(menuScan_5_second,".5 second")
      choice(menuScan_2_second,".2 second")
      choice(menuScan_1_second,".1 second")
}
```

It's not recommended that any of the above choices be deleted. To add more scan periods, such as one for 20 seconds, follow the above format. For example, to add a choice for 0.019 seconds, add the following line after the 0.1 second choice:

```
choice(menuScan_019_second, " .019 second")
```

The range of values for scan periods can be from one clock tick, 0.015 seconds (60 Hz clock), to the maximum number of ticks available on the system. Note, however, that the order of the choices is essential. The first three choices must appear in the above order. Then the remaining choices should follow in descending order, the biggest time period first and the smallest last.

1.2. Event Scanning

There are two types of events supported in the input/output controller (IOC) database, the I/O interrupt event and the user-defined event. For each type of event, the user can specify the scheduling priority of the event using the PRIO or priority field. The scheduling priority refers to the priority the event has on the stack relative to other running tasks. There are three possible choices: `LOW`, `MEDIUM`, or `HIGH`. A low priority event has a priority a little higher than Channel Access. A medium priority event has a priority about equal to the median of periodic scanning tasks. A high priority event has a priority equal to the event scanning task.

I/O Interrupt Events

Scanning on I/O interrupt causes a record to be processed when the I/O card from which it gets its value, generates an interrupt. For example, if an analog input record gets its value from a Xycom 566 Differential Latched card and it specifies I/O interrupt as its scanning routine, then the record will be processed each time the card generates an interrupt (not all types of I/O cards can generate interrupts). Note that even though some cards cannot actually generate interrupts, some driver support modules can simulate interrupts. In order for a record to scan on I/O interrupts, its SCAN field must specify `I/O Intr`.

User-defined Events

The user-defined event mechanism processes records that are meaningful only under specific circumstances. User-defined events can be generated by the `post_event()` database access routine. Two records, the event record and the timer record, are also used to post events. For example, there is the timing output, generated when the process is in a state where a control can be safely changed. Timing outputs are controlled through Timer records, which have the ability to generate interrupts. Consider a case where the timer record is scanned on I/O interrupt and the timer record's event field (EVNT) contains an event number. When the record is scanned, the user-defined event will be posted. When the event is posted, all records will be processed whose SCAN field specifies event and whose event number is the same as the generated event. User-defined events can also be generated through software. Event numbers are configurable and should be controlled through the project engineer. They only need to be unique per IOC because they only trigger processing for records in the same IOC.

All records that use the user-defined event mechanism must specify `Event` in their SCAN field and an event number in their EVNT field.

1.3. Passive Scanning

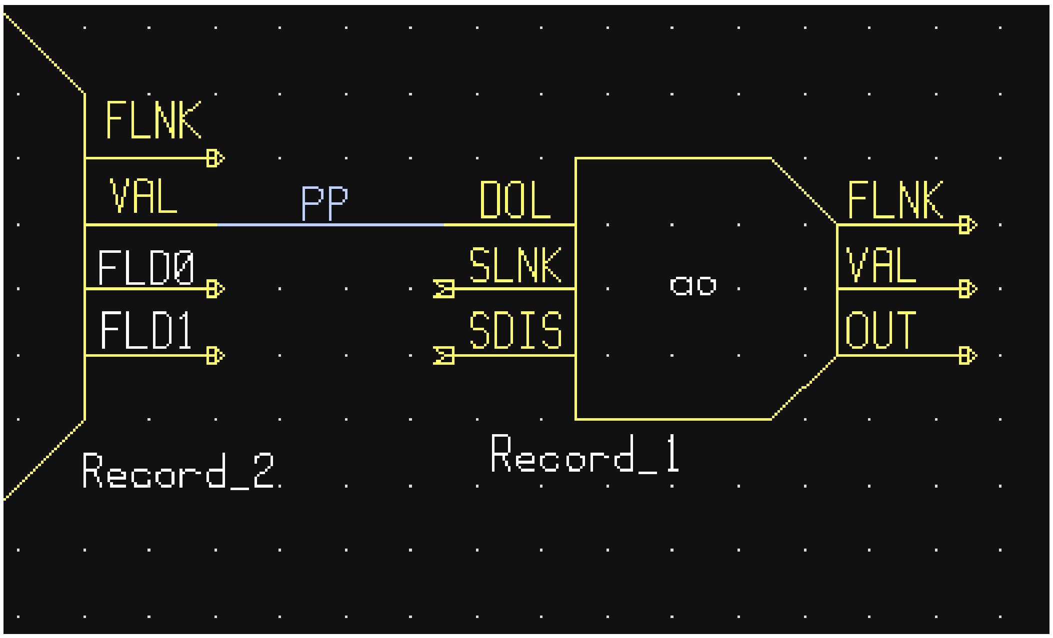
Passive records are records that are processed either when (1) a database access 'put' occurs or when (2) a connecting record is processed. In order for a record to be passive, its SCAN field must specify `Passive`. In case 1, an operator might change the value of a record field using the operator interface, after which the record would process and perform any operations on the data. In case 2, an analog input might retrieve its input from another record. In order for the analog input to retrieve current data, the other record must process its data before the analog input retrieves the data from it. If the other record is a passive record, the analog input record will cause it to process. In case 2, the link field, the INP field in the analog input record for example, must specify process passive (`PP`). Thus, two conditions must be true in case 2:

1. The record referenced by the link must specify `Passive` in its `SCAN` field.
2. The link from the connecting record must specify `PP` or process passive.

Forward links are always process passive, so they cannot specify `NPP`. All input links, on the other hand, have the *process passive* attribute which can be `PP` (process passive True) or `NPP` (no process passive, False). For input links such as INP or DOL (desired output location), if they specify `PP`, they will cause the record whose address they contain to process when the records containing them are processed (When we say a record is specified in the link, we mean an address to a field in that record. When no field is specified in any link, by default the value field `VAL` is accessed. See Section 2, *Address Specification* for more information on how to specify links between records.). This means that the value will be retrieved from the specified record only after the record has processed its data.

For example, *Figure 1* presents a Capfast schematic of two records, Record_1 and Record_2. Record_1 is an analog output record. Most output records have a DOL or Desired Output Link, from which they can retrieve the value that they output. Thus, the DOL link is an input link which can be process passive. The blue line connecting the records is a data link. In this case it means that the DOL link is connected to the VAL field of Record_2. In other words, Record_1 retrieves its value, the value that it outputs, from Record_2. When Record_1 begins processing, it will first retrieve the value from the field connected to DOL, which, in this case, is the VAL field of Record_2. If DOL is process passive, it will cause Record_1 to be processed when the value is retrieved. Record_2 will then process. After Record_1 finishes processing, the value from its VAL field will be retrieved by DOL. Record_1 will then finish its processing.

Figure 1



If NPP is specified as an input link's attribute, the value is retrieved as is from the other record without causing the other record to process. So in the above example, if DOL didn't specify process passive, record 1 would not cause record 2 to process.

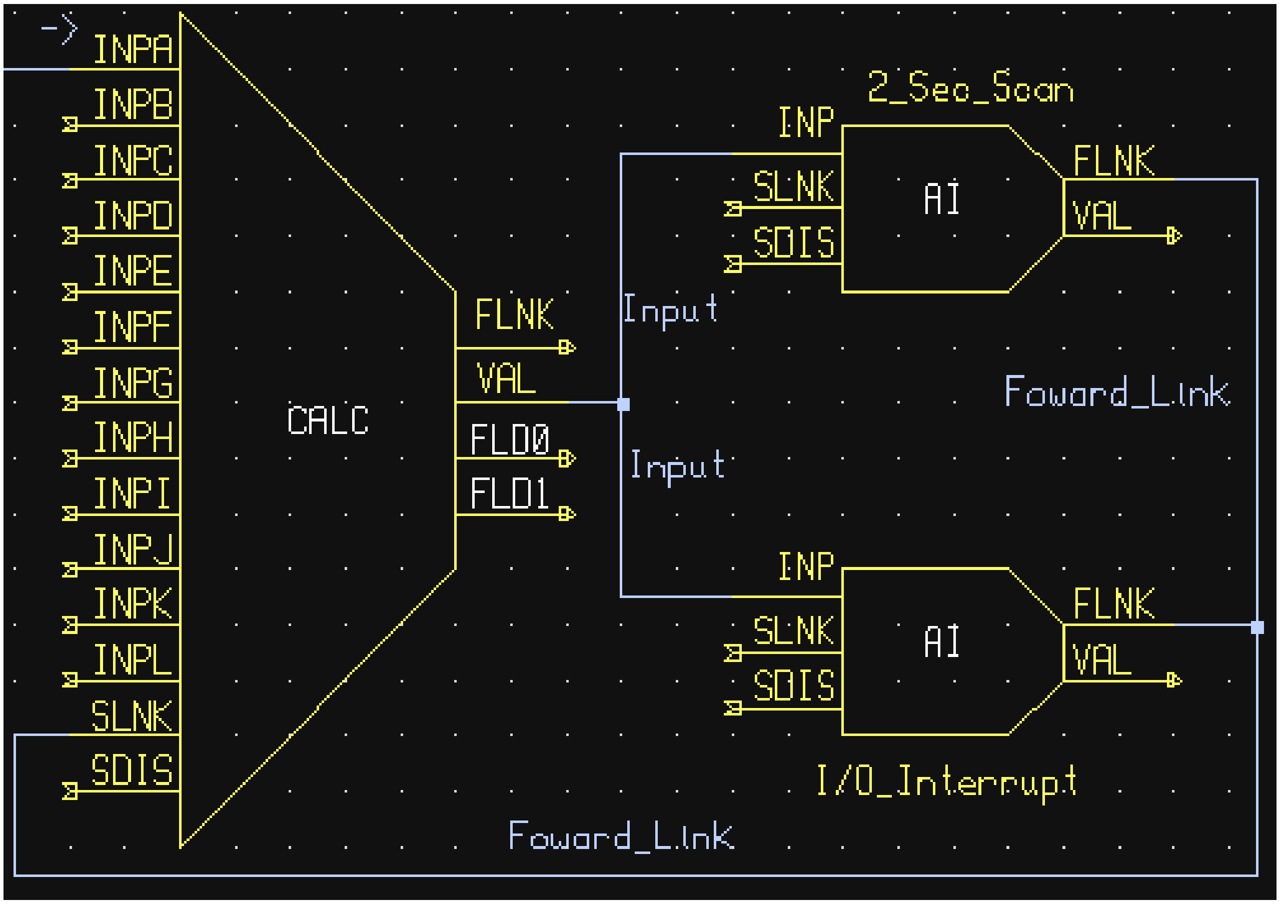
Let's consider a few more examples of passive scanning.

Consider a case where an analog output is controlled only by the operator. There is no reason to process this record until the operator changes the desired output. (This is done by writing to the VAL field.) If this record is passive, the database access routine that places the new desired output into the record will cause it to be processed immediately.

A more complex use of passive scanning causes passive records to inherit the scan traits of the records to which they are connected. Let's look at the simple case (*Figure 2*) where two analog input records (AI) get their input from the VAL field of a calculation record (CALC). Each analog input has a forward link (FLNK) to the calculation record. In Capfast, FLNKs connect to the SLNK field of another record. However, this is just a way to specify the record. In EPICS, there are no SLNK records; a FLNK merely contains the name of another record.

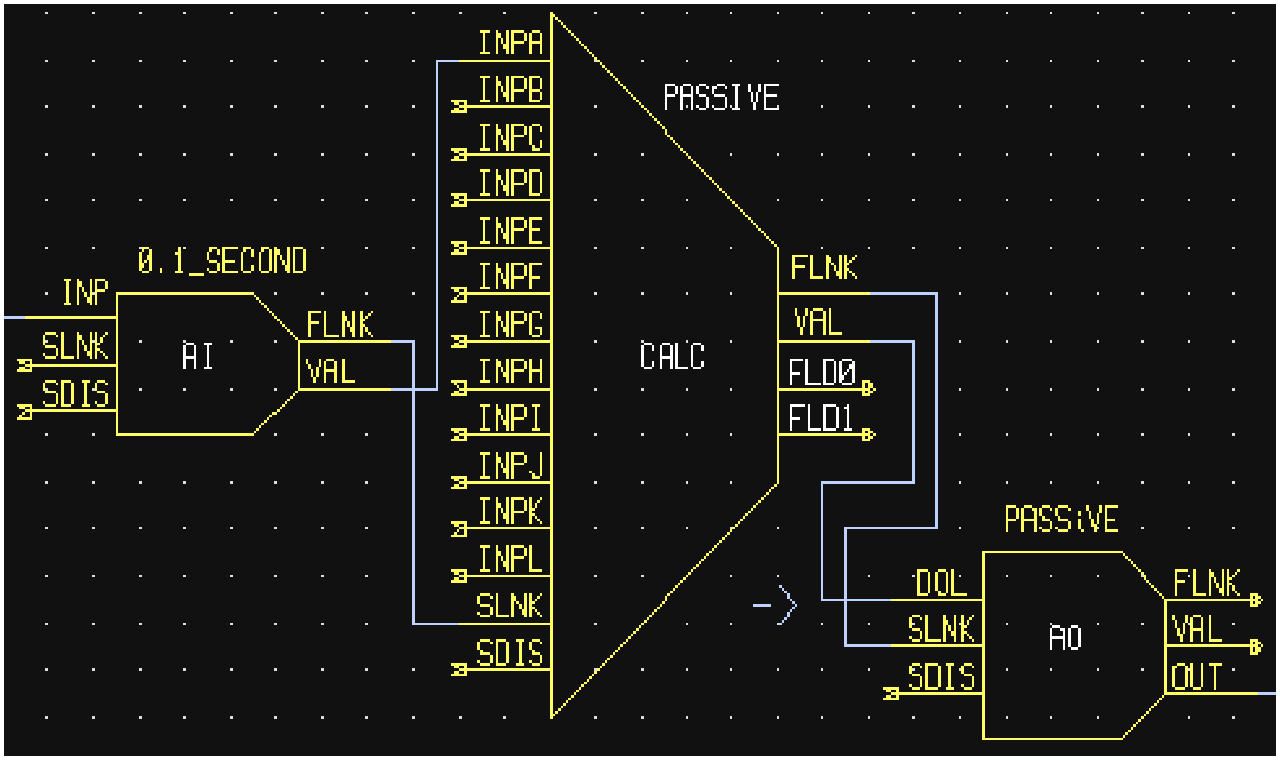
The calculation record's SCAN field specifies `Passive`, the SCAN field of the first analog input record specifies `2 second`, and the SCAN field of the second analog input specifies `I/O Intr`. In this example the calculation will be processed every two seconds and whenever the I/O card interrupts. Thus, the calculation inherits the periodic scanning trait of the first analog input record and the I/O event scanning trait of the second. Each time the calc record is processed, it will retrieve values from the locations specified in its input links and perform its calculation.

Figure 2



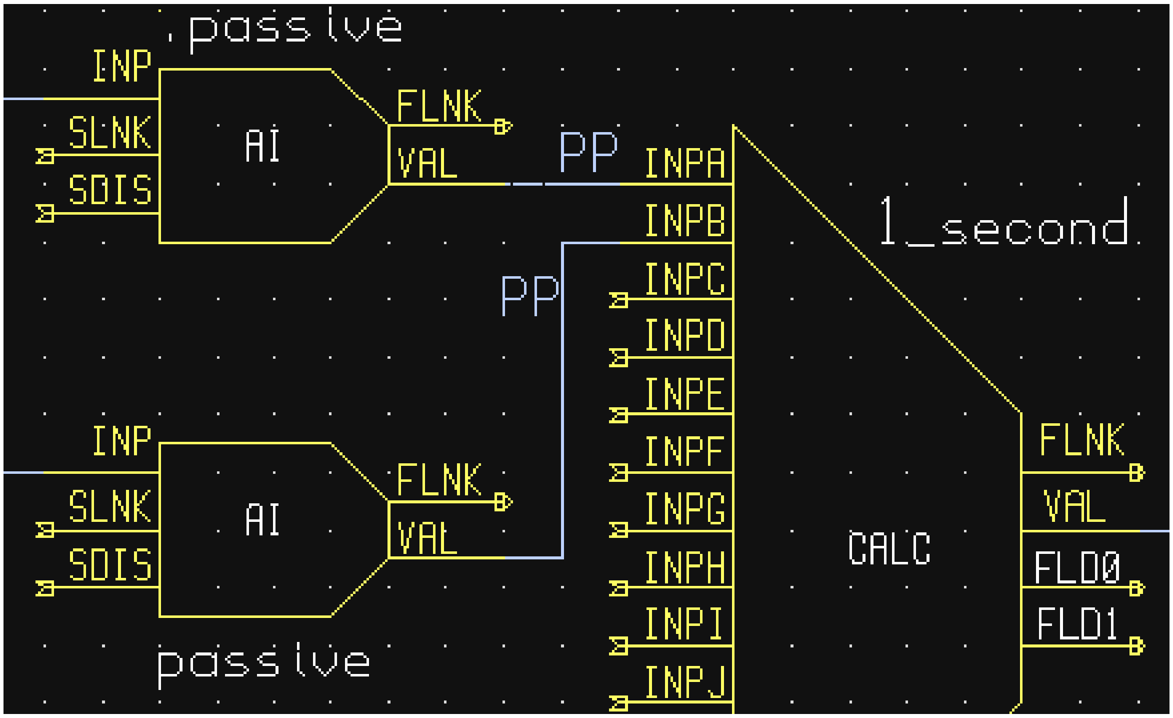
Next, let's look at a continuous control loop (*Figure 3*). In this case the analog input, which is scanned every 0.1 seconds, has a forward link to the calculation record, and the calculation record, in turn, has a forward link to the analog output. Every 0.1 seconds the analog input will process, converting its value and causing the calc record to process. The calc record will make its calculation, causing the analog output record to process. The analog output will then write its output after fetching, if necessary, its desired output. If the operator changes a value in the calculation, this will also cause the calc record to perform its calculation and the analog output to write its output, since the calc and the analog output record are passive.

Figure 3



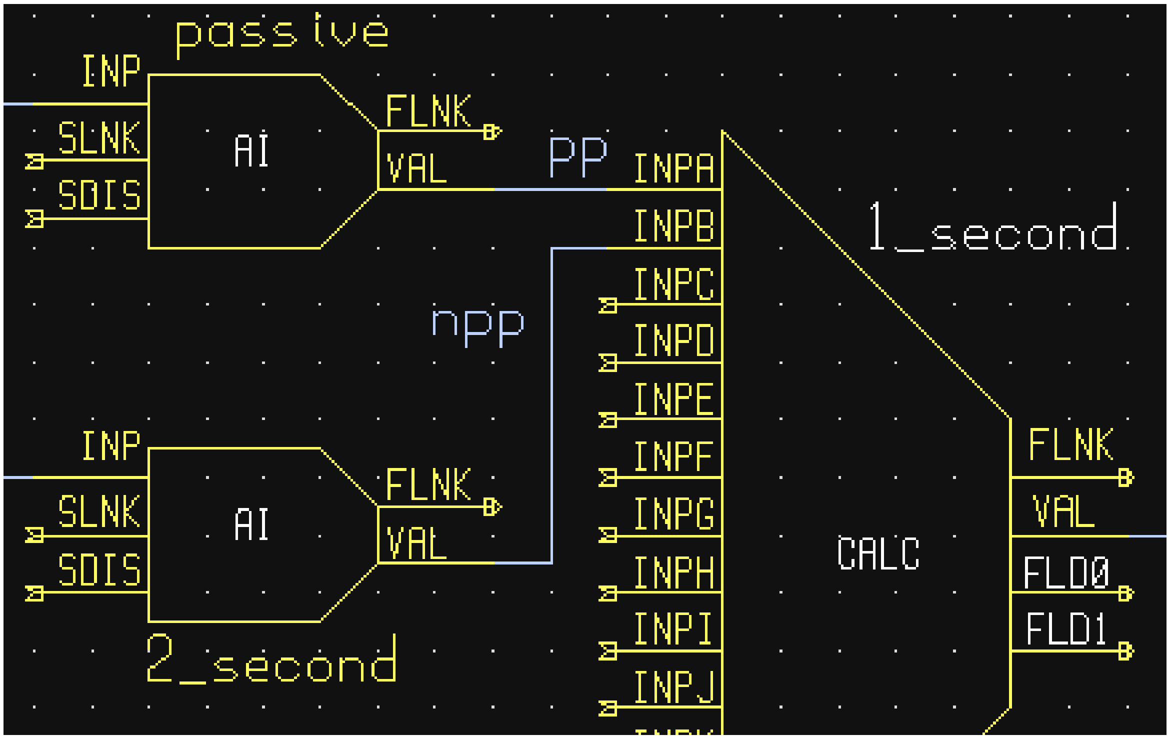
Let's consider a case (*Figure 4*) where values are fetched from other records via input links. When a record fetches a value from another record, the other record is first processed, only if the other record is passive and only if the link specifies process passive or `PP`. As an example, suppose a calculation record has two input links, each of which specifies an analog input record and each of which specifies process passive (`PP`). Suppose also that the calculation record specifies `1 second` in its SCAN field, meaning that it is scanned every second. Every second, the calc record will cause each analog input to process before fetching the values, provided that each analog input specifies `Passive` in its SCAN field.

Figure 4



In a variation of this example (*Figure 5*), suppose one of the analog inputs specifies `2 second` in its SCAN field which means it would no longer be a passive record. Thus, the periodically scanned analog input will *not* be processed every time the calculation is processed. Its current value will simply be fetched as is; then the other analog input will be processed and the calculation performed. The same thing would occur if the calculation's INPB link specified NPP or no process passive. In this case, even if the analog input's SCAN field specified `Passive`, the value would be fetched as is without causing the analog input to process.

Figure 5



Passive Scanning and Channel Access Links

Passive scanning differs somewhat for Channel Access links. A Channel Access link is an input link or output link that specifies a link to a record located in another IOC (a forward processing link can be a CA link under certain circumstances). In addition, as of Release 3.13 input and output links can be forced to be Channel Access links even if they reference a record located in the same database. Input links can specify CA, CP, or CPP. If the input link specifies CA, it will be forced to be a Channel Access link. If the input link specifies CP, it will also be forced to be a Channel Access link; in addition, it will cause the record containing the input link to process whenever a monitor is posted, no matter what the record's SCAN field specifies. If the input link specifies CPP, it means the same thing as CP, except that the record will be processed if and only if the record itself specifies passive in its SCAN field. Output links can specify CA, which will simply cause them to be Channel Access links.

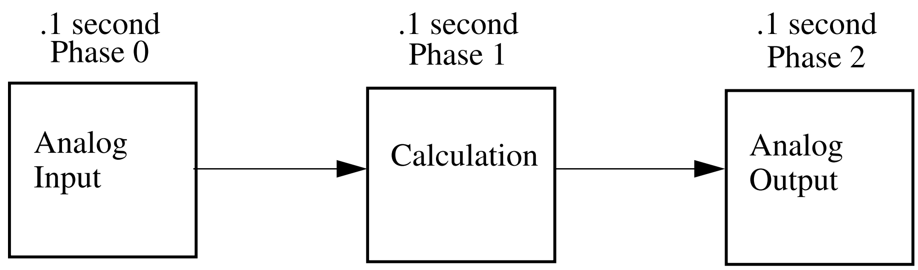
Channel Access links, be they between records located in different IOCs or between records located in the same IOC, cannot be process passive, e.g., they cannot cause the record they specify to process when written to or read from.

1.4. Phase

The PHAS field is used to order the processing of records that are scanned at the same time, i.e., records that are scanned periodically at the same interval and priority, or that are scanned on the same event. In this manner records dependent upon other records can be assured of using current data.

To illustrate this we will look at an example from the previous section, with the records, however, being scanned periodically instead of passively (*Figure 6*). In this example each of these records specifies `.1 second`; thus, the records are synchronous. The phase sequence is used to assure that the analog input is processed first, meaning that it fetches its value from the specified location and places it in the VAL field (after any conversions). Next, the calc record will be processed, retrieving its value from the analog input and performing its calculation. Lastly, the analog output will be processed, retrieving its desired output value from the calc record's VAL field (the VAL field contains the result of the calc record's calculations) and writing that value to the location specified it its OUT link. In order for this to occur, the PHAS field of the analog input record must specify 0, the PHAS field of the calculation record must specify 1, and the analog output's PHAS field must specify 2.

Figure 6



It is important to understand that in the above example, no record causes another to be processed. The phase mechanism instead causes each to process in sequence.

1.5. Forward Process Links

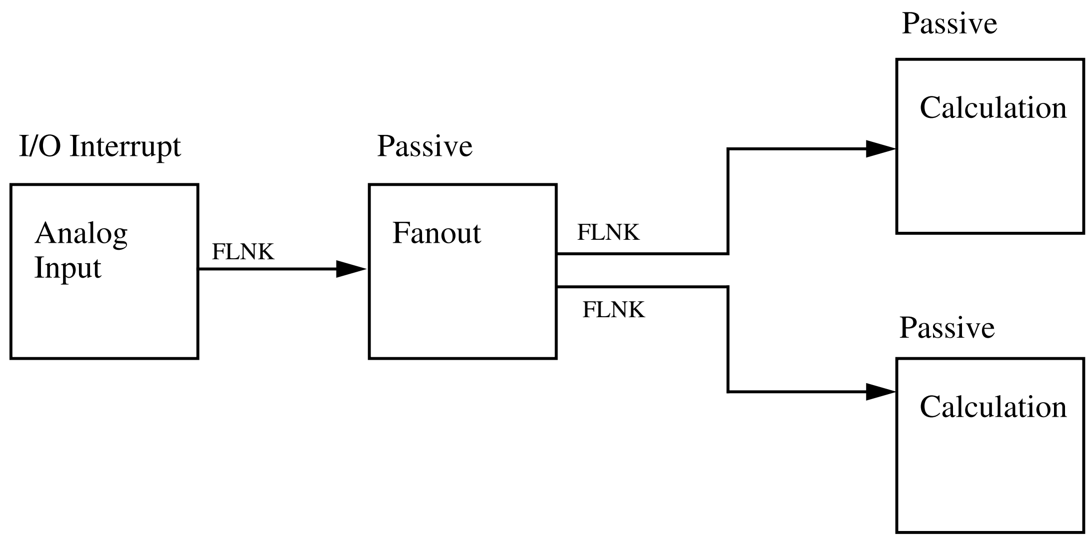
When the *forward processing link* field (FLNK) of one record contains an address of a second record, it causes the second record to be processed after the first record is itself processed.

We discussed forward links in the section on passive processing (*Chapter 1, 1.3*). To reiterate, this field causes the record that it specifies to be scanned when the record that contains the forward link is scanned. It is thus used to cause related records to process. (For more on specifying records in link fields, see *Address Specification, Chapter 1, 2*).

If a forward link references the PROC field of a record in another IOC, a Channel Access " put" request is directed to the specified record, causing it to process.

One record type exists solely to propagate forward processing: the fanout record. The fanout record is used when there is more than one record which needs to be processed as a result of another record being processed. It can specify as many as six forward links. Let's look at an example where an analog input's value is used in two different calculations (*Figure 7*). Because there is only one forward processing link in the analog input record, it is used to process the fanout record. Here two of the fanout records forward links are used to link to two calculation records. In the example, when the I/O interrupt occurs, the analog input is processed, then the fanout record is processed, causing each of the calculation records to be processed. Note that the fanout record simply causes the specified records to process. It does not send values to other records. The *data fanout* record, on the other hand, does send values to other records. Refer to Chapter 13, *Fanout*, and Chapter 11, *dfanout*, for more information on the fanout and data fanout records.

Figure 7



2. Address Specification

Address parameters specify where an input record obtains input, where an output record obtains its desired output values, and where an output record writes its output. They are used to identify links between records, and to specify the location of hardware devices. The most common link fields are OUT, an output link, INP, an input link, and DOL (desired output location), also an input link.

There are three basic types of address specifications which can appear in these fields: hardware addresses, database addresses, and constants.

Note: Not all links support all three types, though some do. However, this doesn't hold true for algorithmic records, which cannot specify hardware addresses. Algorithm records are records like the Calculation, PID, and Select records. These records are used to process values retrieved from other records. Consult the documentation for each record consult.

2.1. Hardware Addresses

Hardware addresses are used to specify input and output connections to hardware devices. They give the information needed by the IOC to interface to the instrumentation. There are currently eight I/O buses supported: VME, Allen-Bradley, CAMAC, GPIB, BITBUS, INST, VXI, and RF. The input specification for each of these is different.

VME Bus

The VME address specification format differs between the various devices. In all of these specifications the '#' character designates a hardware address. The three formats are:

#<Cx Sy @parm> For analog in, analog out, and timer

- C precedes the card number ‘x’
- S precedes the signal number ‘y’
- @ precedes optional string ‘parm’

The card number in the VME addresses refers to the logical card number. Card numbers are assigned by address convention; their position in the backplane is of no consequence. The addresses are assigned by the technician who populates the backplane, with the logical numbers well-documented. The logical card numbers start with 0 as do the signal numbers. *parm* refers to an arbitrary string of up to 31 characters and is device specific.

Allen-Bradley Bus

The Allen-Bradley address specification is a bit more complicated as it has several more fields. The ‘#’ designates a hardware address. The format is:

#<La Ab Cc Sd @parm> All record types

- L precedes the serial link number ‘a’ and is optional - default 0
- A precedes the adapter number ‘b’ and is optional - default 0
- C precedes the card number ‘c’
- S precedes the signal number ‘d’
- @ precedes optional string ‘parm’

The card number for Allen-Bradley I/O refers to the physical slot number, where 0 is the slot directly to the right of the adapter card. The Allen-Bradley I/O has 12 slots available for I/O cards numbered 0 through 11. Allen-Bradley I/O may use double slot addresses which means that slots 0,2,4,6,8, and 10 are used for input modules and slots 1,3,5,7,9 and 11 are used for output modules. It’s required to use the double slot addressing mode when the 1771IL card is used as it only works in double slot addressing mode. This card is required as it provides Kilovolt isolation.

Camac Bus

The CAMAC address specification is similar to the Allen-Bradley address specification. The ‘#’ signifies a hardware address. The format is:

#<Ba Cb Nc Ad Fe @parm> For waveform digitizers

- B precedes the branch number ‘a’
- C precedes the crate number ‘b’
- N precedes the station number ‘c’
- A precedes the subaddress ‘d’ (optional)
- F precedes the function ‘e (optional)
- @ precedes an optional character string ‘parm’

The waveform digitizer supported is only one channel per card; no channel was necessary.

Others

The GPIB, INST, BITBUS, RF, and VXI card-types have been added to the supported I/O cards. A brief description of the address format for each follows. For a further explanation, see the specific documentation on each card.

#<La Ab @parm>   For GPIB I/O

- L precedes the link number ‘a’
- A precedes the GPIB address ‘b’
- @ precedes an optional string of up to 31 characters

#<La Nb Pc Sd @parm> For BITBUS I/O

- L precedes the link ‘a’, i.e., the VME bitbus interface
- N precedes the bitbus node ‘b’

- P precedes the port on node ‘c’
- S precedes the signal on port ‘d’
- @ precedes optional string

#<@ parm> For INST I/O
- @ precedes optional string

#Va Cb Sc @parm For VXI I/O, dynamic addressing
- V precedes the VXI frame number ‘a’
- C precedes the slot within VXI frame ‘b’
- S precedes the signal number ‘c’
- @ precedes optional character string

#Va Sb @parm For VXI I/O, static addressing
- V precedes the logical address ‘a’
- S precedes the signal number ‘b’
- @ precedes optional character string

2.2. Database Addresses

Database addresses are used to specify input links, desired output links, output links, and forward processing links. The format in each case is the same:

```
<RecordName>.<FieldName>
```

where `RecordName` is simply the name of the record being referenced and `FieldName` is the name of the field within the record.

The record name and field name specification are case sensitive. The record name entered in Capfast (or whatever other configuration tool) can be a mix of upper and lower case letters. The field name is always upper case. If no field name is specified as part of an address, the value field (VAL) of the record is assumed. Forward processing links do not need to include the field name because no value is returned when a forward processing link is used; therefore, a forward processing link need only specify a record name.

For inputs and desired output links, the specified record is processed before the value has been read, and for output links the specified record is processed after the value has been written. In the case of the forward processing link, the record being referenced is processed after the record making the link is processed.

Remember that input links such as INP and DOL (desired output location), can specify process passive (`PP`) or no process passive (`NPP`). When a record’s input link specifies a database address, the record specified by the address will process only if the input link specifies process passive and only if the addressed record specifies `Passive` in its SCAN field. If the input link specifies no process passive (NPP), the addressed record will not be processed even if it is a passive record. Because output links such as OUT are always process passive, they always cause the specified record to be processed, provided that the specified record’s SCAN field is configured as `Passive`.

Basic typecast conversions are made automatically when a value is retrieved from another record—integers are converted to floating point numbers and floating point numbers are converted to integers. For example, a calculation record which uses the value field of a binary input will get a floating point 1 or 0 to use in the calculation, because a calculation record’s value fields are floating point numbers. If the value of the calculation record is used as the desired output of a multi-bit binary output, the floating point result is converted to an integer, because multi-bit binary outputs use integers.

Be aware that other types of conversions may not be made when a value is retrieved from another record. Whether it does or doesn’t depends on the record and the device support routine which the record specifies. Most records must specify a device support routine in their DTYP field. Device support routines take care of the specifics of input and output. For such records, there are device support routines for hardware I/O, and other routines for I/O between records. For example, the analog input has many device support routines for input from hardware, and two routines specific for retrieving input from other records: `Soft Channel` and `Raw Soft Channel`. `Raw Soft Channel` retrieves the input value and performs the specified linear conversions on the value (that is, if the record is configured to perform linear conversions). The `Soft Channel` device support routine, on the other hand, reads the value directly into the VAL field and doesn't perform any linear conversions on the value.

Records that use soft device support routines or have no hardware device support routines are called *soft records*. See the chapter on each record for information about that record's device support.

Channel Access Links

> **Note: Link fields can reference records in a different database, that is, a database that resides in a different IOC. Records residing on different IOCs connect through channel access, so any link that refers to a record in another IOC is called a *channel access link*. The format of a channel access link *does not* differ from that of a regular database link.**
>
> **Channel access links are created when the database is initialized. When the initialization routines cannot find the link in the local database, a channel access link is created.**

As of Release 3.13.0, input and output links can also be forced to be Channel Access links, even when they are located in the same IOC. Input links can specify either CA, CP, or CPP. Specifying CA forces the input link to be a Channel Access link. When an input link becomes a Channel Access link, a Channel Access monitor is established on the field and a buffer is allocated for the field using the field type and the element count of the field. In addition to the value of the input link, the alarm status of the link is monitored. Specifying CP or CPP also forces the input link to be a Channel Access link, but in addition, CP or CPP will force the record that contains the link to be processed when a monitor occurs, that is, if the record is process passive.

Output links can also specify CA, in which case they will be forced to be Channel Access links. When an output link becomes a Channel Access link, a buffer is allocated the first time a "put" operation occurs on the record containing the link. Each time a "put" occurs for the record, the data is retrieved from the buffer. And the buffer is updated. The CP and CPP options are not available for output links.

Forward links can also be Channel Access links, either when they specify a record located in another IOC or when they specify the CA attributes. However, forward links will only be made Channel Access links if they specify the PROC field of another record.

Because of the nature of Channel Access links, they cannot be process passive. For example, if an input link that specifies another record in another IOC but also specifies PP, the PP attribute will be ignored. Another aspect of Channel Access links is that they are never placed in the same lock set as the records they link to.

2.3. Constants

Input link fields and desired output location fields can specify a constant instead of a hardware or database address. A constant, which is not really an address, can be an integer value in whatever format (hex, decimal, etc.) or a floating-point value. The value field is initialized to the constant when the database is initialized, and at run-time the value field can be changed by a database access routine. For instance, a constant may be used in an input link of a calculation record. For non-constant links, the calc record retrieves the values from the input links, and places them in a corresponding value field. For constant links, the value fields are initialized with the constant, and the values can be changed by modifying the value field, not the link field. Thus, because the calc record uses its value fields as the operands of its expression, the constant becomes part of the calculation.

When nothing is specified in a link field, it is a NULL link. Before Release 3.13, the value fields associated with the NULL link were initialized with the value of zero. As of Release 3.13, the value fields associated with the links are not initialized

A constant may also be used in the desired output location or DOL field of an output record. In such a case, the initial desired output value (VAL) will be that constant. Any specified conversions are performed on the value before it is written as long as the device support module supports conversions (the `Soft Channel` device support routine does not perform conversions). The desired output value can be changed by an operator at run-time by writing to the value field.

A constant can be used in an output link field, but no output will be written if this is the case. Be aware that this is not considered an error by the database checking utilities.

3. Conversion Specification

Conversion parameters are used to convert transducer data into meaningful data. Discrete signals require converting between levels and states (i.e., on, off, high, low, etc.). Analog conversions require converting between levels and engineering units (i.e., pressure, temperature, level, etc.). These conversions are made to provide operators and application codes with values in meaningful units.

The following sections discuss these types of conversions. The actual field names appear in capital letters.

3.1. Discrete Conversions

The most simple type of discrete conversion would be the case of a discrete input that indicates the on/off state of a device. If the level is high it indicates that the state of the device is on. Conversely, if the level is low it indicates that the device is off. In the database, parameters are available to enter strings which correspond to each level, which, in turn, correspond to a state (0,1). By defining these strings, the operator is not required to know that a specific transducer is on when the level of its transmitter is high or off when the level is low. In a typical example, the conversion parameters for a discrete input would be entered as follows:

Zero Name (ZNAM): Off

One Name (ONAM): On

The equivalent discrete output example would be an on/off controller. Let's consider a case where the safe state of a device is `On`, the zero state. The level being low drives the device on, so that a broken cable will drive the device to a safe state. In this example the database parameters are entered as follows:

Zero Name (ZNAM): On

One Name (ONAM): Off

By giving the outside world the device state, the information is clear. Binary inputs and binary outputs are used to represent such on/off devices.

A more complex example involving discrete values is one that has many states such as a multi-bit binary output record. Consider a motor which has four states—off, low, medium, and high. A device of this type may have three control lines and three more monitor lines. Each line represents one of the on states (low, medium, or high). The bit pattern for each control state is entered into the database with the string that describes that state. The database parameters for the monitor would be entered as follows:

Number of Bits (NOBT): 3

First Input Bit Spec (INP): Address of the least significant bit

Zero Value (ZRVL): 0

One Value (ONVL): 1

Two Value (TWVL): 2

Three Value (THVL): 4

Zero String (ZRST): Off

One String (ONST): Low

Two String (TWST): Medium

Three String (THST): High

In this case, when the database record is scanned, the monitor bits are read and compared with the bit patterns for each state. When the bit pattern is found, the device is set to that state. For instance, if the three monitor bits read equal `0100` (4), the three value is the corresponding value, and the device would be set to state 3 which drives the device to its high level. The value can be displayed as an integer, in which case the value would be 3, or as a string, in which case the value would be 'High'.

If the bit pattern is not found, the device is in an unknown state. For instance, if the three monitor bits read equal `0111` (7) and there are no equivalent values, then the value is set to -1, the condition of the record is set to UNKNOWN alarm, and the alarm severity is set to whatever alarm severity is configured for the unknown state (see *Alarm Specification, Chapter 1, 4*).

In addition, the DOL fields of binary output records (bo and mbbo) will accept values in strings. When they retrieve the string or when the value field is given a string via `put_enum_strs`, a match is sought with one of the states. If a match is found, the value for that state is written.

3.2. Analog Conversions

Analog conversions require knowledge of the transducer, the filters, and the I/O cards. Together they measure the process, transmit the data, and interface the data to the IOC. Smoothing is available to filter noisy signals. The smoothing argument is a constant between 0 and 1 and is specified in the SMOO field. It is applied to the converted hardware signal as follows:

eng. units = (new eng units * (1 - smoothing) + (prev. eng units * smoothing))

The analog conversions from raw values to engineering units can be either linear or breakpoint conversions.

Whether an analog record performs linear conversions, breakpoint conversions, or no conversions at all depends on how the record's LINR field is configured. The possible choices for the LINR field are as follows:

LINEAR
NO CONVERSION
typeKdegF
typeKdegC
typeJdegF
typeJdegC

If LINEAR is chosen, the record performs a linear conversion on the data. If NO CONVERSION is chosen, the record performs no conversion on its data. The other choices are the names of breakpoint tables. When one of these is specified in the LINR field, the record uses the specified table to convert its data. (Note that additional breakpoint tables are often added at specific sites, so more breakpoint tables than are listed here may be available at the user's site.) The following sections explain linear and breakpoint conversions.

Linear Conversions

There are three formulas to know when considering the linear conversion parameters. The conversion from measured value to engineering units is as follows:

$$\text{eng units} = \left(\frac{\text{measured A/D counts}}{\text{full scale A/D counts}}\right) \times \langle \text{eng units full scale} - \text{eng units low} \rangle + \text{eng units low}$$

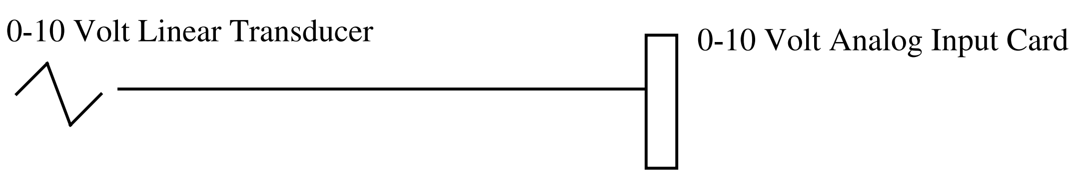
The engineering units full scale and low scale are specified in the EGUF and EGUL fields, respectively. The values of the EGUF and EGUL fields correspond to the maximum and minimum values of the transducer, respectively. Thus, the value of these fields is device dependent. For example, if the transducer has a range of -10 to +10 volts, then the EGUF field should be 10 and the EGUL field should be -10.

In the following examples the determination of engineering units full scale and low scale is shown. The conversion to engineering units is also shown to familiarize the reader with the signal conversions from signal source to database engineering units.

> **Note: The engineering units field (EGU) in an analog record has nothing to do with the conversions. The EGU field simply contains a string that should describe the engineering units used by the record, such as PSI for an analog input that reads values from a device that transmits pressure. Thus, the EGU field is meant for the operator's sake.**

Transducer Matches the I/O module

First let us consider a linear conversion. In this example, the transducer transmits 0-10 Volts, there is no amplification, and the I/O card uses a 0-10 Volt interface.



The transducer transmits pressure: 0 PSI at 0 Volts and 175 PSI at 10 Volts. The engineering units full scale and low scale are determined as follows:

```
eng. units full scale = 17.5 X 10.0
eng. units low scale = 17.5 X 0.0
```

The field entries in an analog input record to convert this pressure will be as follows:

LINR:	Linear
EGUF:	175.0
EGUL:	0
EGU:	PSI

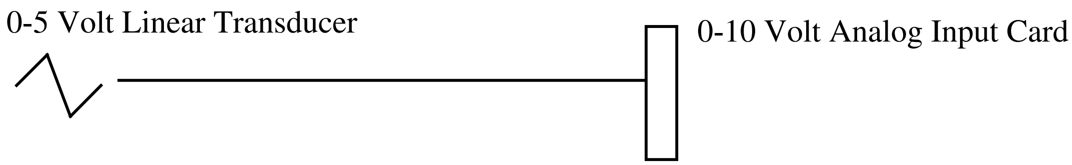
The conversion will also take into account the precision of the I/O module. In this example (assuming a 12 bit analog input card) the conversion is as follows:

$$\text{eng units} = \left(\frac{\text{measured A/D counts}}{4095}\right) \times (175 - 0) + 0$$

When the pressure is 175 PSI, 10 Volts is sent to the I/O module. At 10 Volts the signal is read as 4095. When this is plugged into the conversion, the value is 175 PSI.

Transducer Lower than the I/O module

Let's consider a variation of this linear conversion where the transducer is 0-5 Volts.



In this example the transducer is producing 0 Volts at 0 PSI and 5 Volts at 175 PSI. The engineering units full scale and low scale are determined as follows:

```
eng. units low scale = 35 X 10
eng. units full scale = 35 X 0
```

The field entries in an analog record to convert this pressure will be as follows:

LINR:	Linear
EGUF:	350
EGUL:	0
EGU:	PSI

The conversion will also take into account the precision of the I/O module. In this example (assuming a 12 bit analog input card) the conversion is as follows:

$$\text{eng units} = \left(\frac{\text{measured A/D counts}}{4095}\right) \times (350 - 0) + 0$$

Notice that at full scale the transducer will generate 5 Volts to represent 175 PSI. This is only half of what the input card accepts; input is 2048. Let's plug in the numbers to see the result:

```
(2048 / 4095) * (350 - 0) + 0 = 175
```

In this example we had to adjust the engineering units full scale to compensate for the difference between the transmitter and the analog input card.

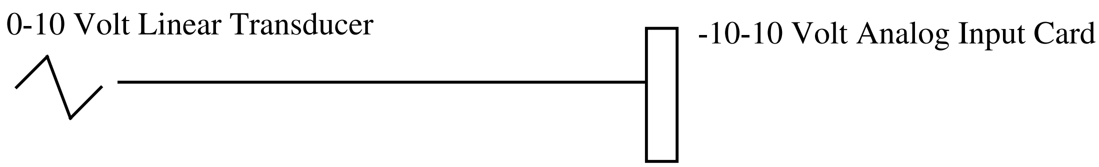
Transducer Positive and I/O module bipolar

Let's consider another variation of this linear conversion where the input card accepts -10 Volts to 10 Volts (i.e. Bipolar instead of Unipolar).

In this example the transducer is producing 0 Volts at 0 PSI and 10 Volts at 175 PSI. The input module has a different range of voltages and the engineering units full scale and low scale are determined as follows:

```
eng. units full scale = 17.5 X 10
eng. units low scale = 17.5 X (-10)
```

The database entries to convert this pressure will be as follows:



LINR:	Linear
EGUF:	175
EGUL:	-175
EGU:	PSI

The conversion will also take into account the precision of the I/O module. In this example (assuming a 12 bit analog input card) the conversion is as follows:

$$\text{eng units} = \left(\frac{\text{measured A/D counts}}{4095}\right) \times (175 - (-175)) + (-175)$$

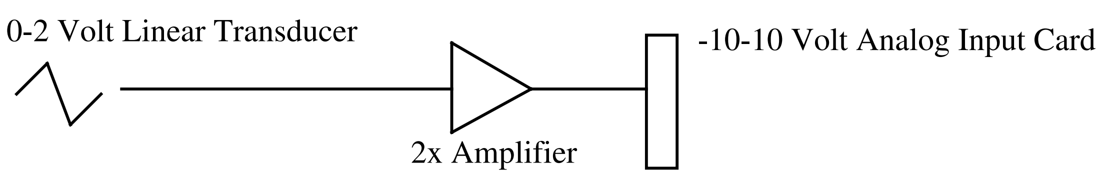
Notice that at low scale the transducer will generate 0 Volts to represent 0 PSI. Because this is half of what the input card accepts, it is input as 2048. Let's plug in the numbers to see the result:

```
(2048 / 4095) * (175 - -175) - 175 = 0
```

In this example we had to adjust the engineering units low scale to compensate for the difference between the unipolar transmitter and the bipolar analog input card.

Combining Linear Conversion with an Amplifier

Let's consider another variation of this linear conversion where the input card accepts -10 Volts to 10 Volts, the transducer transmits 0 - 2 Volts for 0 - 175 PSI and a 2x amplifier is on the transmitter.



At 0 PSI the transducer transmits 0 Volts. This is amplified to 0 Volts. At half scale, it is read as 2048. At 175 PSI, full scale, the transducer transmits 2 Volts, which is amplified to 4 Volts. The analog input card sees 4 Volts as 70 percent of range or 2867 counts. The engineering units full scale and low scale are determined as follows:

```
eng units full scale = 43.75 * 10
eng units low scale = 43.75 * (-10)
```

(175 / 4 = 43.75) The record's field entries to convert this pressure will be as follows:

LINR:	Linear
EGUF:	437.5
EGUL:	-437.5
EGU:	PSI

The conversion will also take into account the precision of the I/O module. In this example (assuming a 12 bit analog input card) the conversion is as follows:

$$\text{eng units} = \left(\frac{\text{measured A/D counts}}{4095}\right) \times (437.5 - (-437.5)) + (-437.5)$$

Notice that at low scale the transducer will generate 0 Volts to represent 0 PSI. Because this is half of what the input card accepts, it is input as 2048. Let's plug in the numbers to see the result:

```
(2048 / 4095) * (437.5 - -437.5) - 437.5 = 0
```

Notice that at full scale the transducer will generate 2 volts which represents 175 PSI. The amplifier will change the 2 Volts to 4 Volts. 4 Volts is 14/20 or 70 percent of the I/O card's scale. The input from the I/O card is therefore 2866 (i.e., 0.7 * 4095). Let's plug in the numbers to see the result:

```
(2866 / 4095) * (437.5 - -437.5) - 437.5 = 175 PSI
```

We had to adjust the engineering units full scale to adjust for the difference between the transducer with the amplifier affects and the range of the I/O card. We also adjusted the low scale to compensate for the difference between the unipolar transmitter/amplifier and the bipolar analog input card.

Breakpoint Conversions

Now let us consider a non-linear conversion, otherwise known as a breakpoint conversion. In this example the transducer is a thermocouple which transmits 0-20 milliAmps. An amplifier is present which amplifies milliAmps to volts. The I/O card uses a 0-10 Volt interface.

The transducer is transmitting temperature. The database entries in the analog input record that are needed to convert this temperature will be as follows:

LINR:	typeJdegC
EGUF:	0
EGUL:	0
EGU:	DGC

For analog records that use breakpoint tables, the EGUF and EGUL fields are not used in the conversion, so they do not have to be given values. Instead, this conversion is completed by performing a table lookup. The value read from the device is between 0 and 4095 and is known as the *raw value*, which is initially read into the RVAL (raw value) field. This raw value is then used to identify the line segment in which this value falls. Each entry of the table includes a beginning point for the segment, the floating engineering units value at the point, and the slope of the line segment. The conversion to the engineering units is as follows:

```
final value = eng. units + (raw value - beginning point) X slope
```

There are currently lookup tables for the J and K thermocouples in degrees F and degrees C.

Other applications for a lookup table are the remainder of the thermocouples, logarithmic output controllers, and exponential transducers. We have chosen the piece-wise linearization of the signals to perform the conversion, as they provide a mechanism for conversion that minimizes the amount of floating point arithmetic required to convert non-linear signals. Additional breakpoint tables can be added to the existing breakpoints.

There are two ways to create a new breakpoint table:

(1) Simply type in the beginning point for each segment and the corresponding engineering units at that point in the following format.

```
breaktable(<tablename>) {
      <beginning point> <eng units>
      ibid.
}
```

where the `<tablename>` is the name of the table, such as typeKdegC, and `<beginningpoint>` is the value of the beginning point for each line segment, and `<eng units>` is the corresponding engineering units. The slope is calculated by the software and should not be specified.

(2) Create a file consisting of a table of an arbitrary number of values in engineering units and use the utility called **makeBpt** to convert the table into a breakpoint table. As an example, the contents data file to create the typeJdegC breakpoint table look like this:

```
!header
"typeJdegC" 0 0 700 4095 .5 -210 760 1
!data
-8.096 -8.076 -8.057 many more numbers
```

The file must have the extension `.data`. The file must first have a header specifying these nine things:

1. Name of breakpoint table in quotes: **"typeJdegC"**
2. Engineering units for 1st breakpoint table entry: **0**
3. Raw value for 1st breakpoint table entry: **0**
4. Highest value desired in engineering units: **700**
5. Raw value corresponding to high value in engineering units: **4095**
6. Allowed error in engineering units: **.5**
7. Engineering units corresponding to first entry in data table: **-210**
8. Engineering units corresponding to last entry in data table: **760**
9. Change in engineering units between data table entries: **1**

The rest of the file contains lines of equally spaced engineering values, with each line no more than 160 characters before the new-line character. The header and the actual table should be specified by !header and !data, respectively. The file for this data table is called `typeJdegC.data`, and can be converted to a breakpoint table with the **makeBpt** utility as follows:

```
%makeBpt TypeJdegC.data
```

4. Alarm Specification

There are two elements to an alarm condition: the alarm *status* and the *severity* of that alarm. Each database record contains its current alarm status and the corresponding severity for that status. The scan task which detects these alarms is also capable of generating a message for each change of alarm state. The types of alarms available fall into these categories: scan alarms, read/write alarms, limit alarms, and state alarms. Some of these alarms are configured by the user, and some are automatic which means that they are called by the record support routines on certain conditions, and cannot be changed or configured by the user.

Alarm Severity

An alarm *severity* is used to give weight to the current alarm status. There are four severities:

1. MINOR
2. MAJOR
3. NO_ALARM
4. INVALID

An alarm state that needs attention but is not dangerous is a MINOR alarm. In this instance the alarm state is meant to give a warning to the operator. A serious state is a MAJOR alarm. In this instance the operator should give immediate attention to the situation and take corrective action. NO_ALARM means no alarm has been triggered. An INVALID alarm means there's a problem with the data, which can be any one of several problems; for instance, a persons trips over some writes, unplugging them, which disables the sensors, which means that a new value couldn't be scanned for the record. In that case, an alarm of INVALID severity will be triggered. When a system is being tested, an INVALID alarm can point to a simply configuration problem. However, when the system is actually on-line, an INVALID alarm can signal a much more serious problem.

For limit alarms and state alarms, the severity can be configured by the user to be MAJOR or MINOR for the a specified state. For instance, an analog record can be configured to trigger a MAJOR alarm when its value exceeds 175.0. In addition to the MAJOR and MINOR severities, the user can choose the NO_ALARM severity, in which case no alarm is generated for that state.

For the other alarm types (i.e., scan, read/write), the severity is always INVALID and not configurable by the user. An INVALID alarm

Scan Alarm

A scan alarm is generated if a record is not successfully placed in the desired scan list, or if it is found by the scan task to be locked in ten successive attempts to process it. When a scan alarm occurs, the alarm severity is always set to INVALID.

Read Alarm

This alarm status is returned when the value is fetched from hardware or from a database field. If the read routine fails, the READ_ALARM condition exists. The severity of this alarm condition is always set to INVALID.

Write Alarm

This alarm status is returned when the value is written either to hardware or to a database field. If the write fails, the WRITE_ALARM condition exists. The severity of this alarm condition is always set to INVALID.

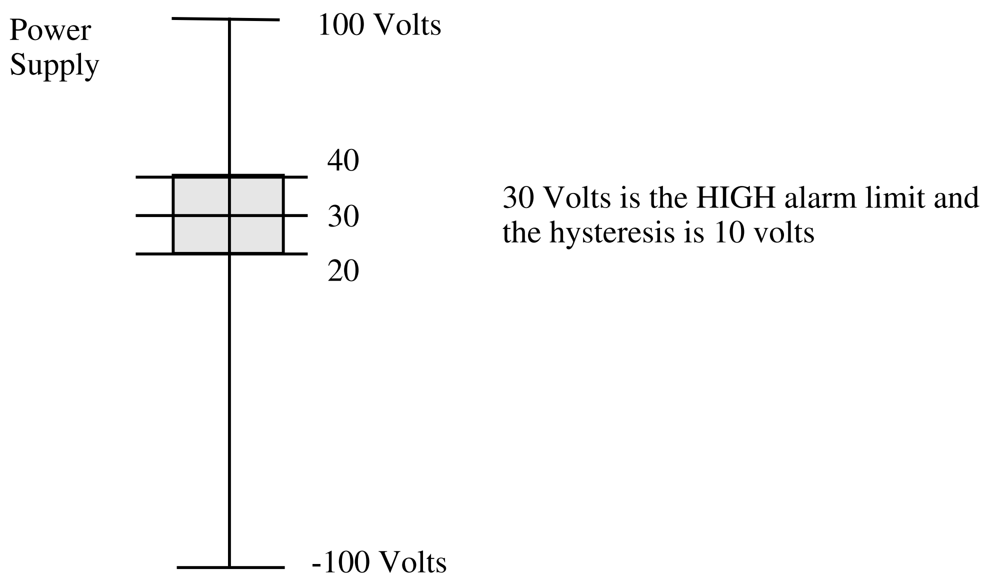
Limit Alarms

For analog records (this includes such records as the stepper motor record), there are configurable alarm limits. There are two limits for above normal operating range and two limits for the below-limit operating range. Each of these limits has an associated alarm severity which is configured in the database. If the record's value drops below the low limit and an alarm severity of MAJOR was specified for that limit, then a MAJOR alarm is triggered. **When the severity of a limit is set to NO_ALARM, none will be generated—even if the limit entered has been violated.**

There are two limits at each end—two low values and two high values—so that a warning can be set off before the value goes into a dangerous condition.

Analog records also contain a hysteresis field, which is also used when determining limit violations. The hysteresis field is the deadband around the alarm limits. The deadband keeps a signal that is hovering at the limit from generating too many alarms. Let's take an example (*Figure 8*) where the range is -100 to 100 volts, the high alarm limit is 30 Volts, and the hysteresis is 10 Volts. If the value is normal and approaches the HIGH alarm limit, an alarm is generated when the value reaches 30 Volts. This will only go to normal if the value drops below the limit by more than the hysteresis. For instance, if the value changes from 30 to 28 this record will remain in HIGH alarm. Only when the value drops to 20 will this record return to normal state.

Figure 8



State Alarms

For discrete values there are configurable state alarms. In this case a user may configure a certain state to be an alarm condition. Let's consider a cooling fan whose discrete states are high, low, and off. The off state can be configured to be an alarm condition so that whenever the fan is off the record is in a STATE alarm. The severity of this error is configured for each state. In this example, the low state could be a STATE alarm of MINOR severity, and the off state a STATE alarm of MAJOR severity.

Discrete records also have a field in which the user can specify the severity of an unknown state to NO_ALARM, MINOR or MAJOR. Thus, the unknown state alarm is not automatic.

Discrete records also have a field which can specify an alarm when the record's state changes. Thus, an operator can know when the record's alarm state has changed. If this field specifies NO_ALARM, then a change of state will not trigger a change of state alarm. However, if it specifies either MINOR or MAJOR, a change of state will trigger an alarm with the corresponding severity.

Alarm Handling

A record handles alarms with the NSEV, NSTA, SEVR, and STAT fields. When a software component wants to raise an alarm, it first checks the new alarm state fields: NSTA, new alarm state, and NSEV, new alarm severity. If the severity in the NSEV field is higher than the severity in the current severity field (SEVR), then the software component sets the NSTA and NSEV fields to the severity and alarm state that corresponds to the outstanding alarm. When the record process routine next processes the record, it sets the current alarm state (STAT) and current severity (SEVR) to the values in the

NSEV and NSTA fields. This method of handling alarms ensures that the current severity (STAT) reflects the highest severity of outstanding alarm conditions instead of simply the last raised alarm. This also means that the if multiple alarms of equal severity are present, the alarm status indicates the first one detected.

In addition, the `get_alarm_double()` routine can be called to format an alarm message and send it to an alarm handler. The alarm conditions may be monitored by the operator interface by explicitly monitoring the STAT and SEVR fields. All values monitored by the operator interface are returned from the database access with current status information.

5. Monitor Specification

Monitors are a mechanism that provide a user program with data from the database without the user having to constantly poll the database. Through channel access, monitors inform the operator interface, archivers, alarm handlers and other user programs when a database field changes. Monitors can be placed on any field that can be accessed through the database access layer: floats, integers, strings, enumerated, and link fields. The fields involved with monitoring fall into two categories: determining when to notify a user and maintaining the list of monitors. For more information about using monitors, see the Channel Access Reference Guide.

5.1. Notification

For most fields that are accessible through the database access layer, users are notified whenever the field changes. The exception is the VAL or value field found in most records. Monitors on the value fields are sent when either the value changes or the alarm condition changes. Value fields of the floating-point type are special in that there are two deadbands around the monitor notification: one for archive monitors, ADEL, and one for all other monitors, MDEL. These deadbands are provided to aid the user in reducing the amount of processing by filtering out negligible value changes. These numbers should be set after considering the precision required by the application. Setting these deadbands carefully could considerably extend the capability of an I/O Controller.

To implement the deadbands, each record that has deadbands for the value field or fields (not all records have deadbands for value fields) will have fields that contain the value for the monitored field from the last time the record was processed. For instance, an analog output has the ALST and MLST fields. The first implements the deadband for the archivers; the second, for all other monitors on the value field. Each time the record is processed, the last value is compared to the current value, and if the change is greater than the deadband, monitors for the field are sent.

```
if ((current value - last value) > deadband)
     send monitors
```

Of course, the formula is a little bit more complicated in order to deal with negative numbers and other subtleties, but the basic idea is the same.

5.2. List Maintenance

Each record keeps track of all the monitors that are active as a result of Channel Access monitor requests. A Channel Access monitor request occurs when a client has requested to monitor a specific record or field. The head of the list of monitors for a record currently active is found in the monitor list field (MLIS). Monitors are active when the value of MLIS is greater than 0.

6. Control Specification

Closed loop control is achieved by linking output records to other records. In all output records there are a set of fields available to specify a location from which the desired output should come. After the desired output value is retrieved, it is converted and then written. Usually, these fields are called DOL, or desired output location field, and the OMSL, or output mode select field. The OMSL field has two mode choices: `closed_loop or supervisory`. When the closed loop mode is chosen, the desired output is retrieved from the location specified by the DOL field and placed into the VAL field. When the supervisory mode is chosen, the desired output value is simply taken from VAL and is not retrieved from DOL. In the supervisory mode, VAL can be changed externally via the database access routines.

The DOL field, like most other link fields, can be a link or a constant. When a constant, the VAL field is initialized to its value. Thus, if the desired output of the record is to be controlled externally by the operator, the DOL field can be given a value with which the VAL field will be initialized.

When an output record is set to closed loop mode, new desired outputs can be retrieved from algorithm records so that the output required can achieve a desired affect. Using this capability, closed control loops can be implemented. A closed control loop usually consists of an input record which retrieves a value from a process variable, an algorithm record which retrieves its value from the input record and manipulates the data in some way, and then an output record which retrieves the changed value from the algorithm record and sends out to a controller that will determine the input. This section will look at an example of closed loop control.

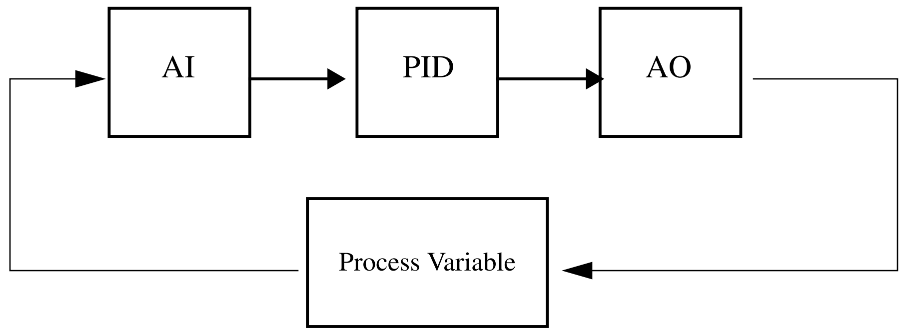
6.1. Closing an Analog Control Loop

In a simple control loop an analog input record reads the value of a process variable or PV. Then, a PID record retrieves the value from the analog input record. A PID record reads a value, determine if the value is in error, and if so, corrects the error to conform with the desired output. Then this value can be used an output record to write the value to the same PV. In this case, an analog output record gets the value from the PID and writes it to the PV. In the database, the PID record gets the current value from the analog input record, and the analog output record gets the desired output from the PID record.

For a closed control loop, `closed_loop` should be chosen in the analog output's OMSL field, in which case the desired output is retrieved from the DOL field which should specify the DM field of the PID record. The PID record determines the difference between the desired value and the current process value gotten from the analog input, and also determines how the output should change to get the process value to the desired value. The result of the PID calculation is an output delta. The analog output needs to be configured to accept incremental changes when tied to a PID that calculates change only. This can be done in the OIF field, which has two choices: `Full` or `Incremental`.

The analog output record, like most output records, can be taken out of the closed control loop by changing `closed_loop` to `supervisory` in its OMSL field, which is accessible and modifiable at run-time. Doing this, of course, suspends the closed control loop until the OMSL field is set back to `supervisory`,

Figure 9



Chapter 2: Fields Common to All Record Types

1. Introduction

This chapter contains a description of fields that are common to all records. These fields are defined in dbcommon.dbd.

2. Scan Fields

These fields contain information related to how and when a record processes. For a further explanation of these record processing and these fields, see *Scanning Specification, Chapter 1, 1*. A few records have unique fields that also affect how they process. These fields, if any, will be listed and explained in the chapter for each record.

2.1. Field Summary

Field	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SCAN	GBLCHOICE	Yes	0	Yes	Yes	No	No
PINI	GBLCHOICE	Yes	0	Yes	Yes	No	No
PHAS	SHORT	Yes	0	Yes	Yes	No	No
EVNT	SHORT	Yes	0	Yes	Yes	No	No
PRIO	GBLCHOICE	Yes	0	Yes	Yes	No	No
DISV	SHORT	Yes	1	Yes	Yes	No	No
DISA	SHORT	No	0	Yes	Yes	No	No
SDIS	INLINK	Yes	0	No	No	N/A	No

Field	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
PROC	UCHAR	No	0	Yes	Yes	No	Yes
DISS	GBLCHOICE	Yes	0	Yes	Yes	No	No
LSET	SHORT	No	0	Yes	No	No	
LCNT	UCHAR	No	0	Yes	No	No	
PACT	UCHAR	No	0	Yes	No	No	
FLNK	FWDLINK	Yes	Null	No	No	N/A	No

2.2. Field Description

Name	Summary	Description
SCAN	Scanning Algorithm	This can be one of the periodic intervals (`.1 second`, `.2 second`, `.5 second`, `1 second`, `2 second`, `5 second`, `10 second`, `I/O Intr`, `Event`, or `Passive`.
PINI	Process at Initialization	If this field is set to TRUE during database configuration, then the record is processed once at IOC initialization (before the normal scan tasks are started).
PHAS	Scan Phase Number	This field orders the records within a specific SCAN group. This is not meaningful for passive records. All records of a specified phase are processed before those with higher phase number. Whenever possible it is better to use linked passive records to enforce the order of processing rather than phase number.
EVNT	Event Number	Event number for scan type SCAN_EVENT. All records with scan type event and the same EVNT value will be processed when a call to post_event for EVNT is made. The call to post_event is: post_event(short event_number)
PRIO	Priority	Scheduling priority for processing I/O Event scanned records and asynchronous record completion tasks.
DISV	Disable Value	If DISV=DISA, then the record will be disabled, i.e. dbProcess will not process the record.
DISA	Scan Disable Input Link Value	This is the value that is compared with DISV to determine if the record is disabled. Its value is obtained via SDIS if SDIS is a database or channel access link. If SDIS is not a database or channel access link, then DISA can be set via dbPutField or dbPutLink.
SDIS	Scan Disable Input Link	An input link from which to obtain a value for DISA. This field is ignored unless it is a database link or a channel access link. If it is a database or a channel access link, dbProcess calls dbGetLink to obtain a value for DISA before deciding to call the processing routine.
PROC	Process Record	A record will be processed whenever a dbPutField is directed to this field.

Name	Summary	Description
DISS	Disable Alarm Severity	When this record is disabled, it will be put into alarm with this severity and a status of DISABLE_ALARM.
LSET	Lock Set	The lock set to which this record belongs. All records linked in any way via input, output, or forward database links belong to the same lock set. The only exception is that non-process passive input links do not force the linked record to be in the same lock set. The lock sets are determined at IOC initialization time.
LCNT	Lock Count	The number of times in succession dbProcess finds the record active, i.e. PACT is TRUE. If dbProcess finds the record active MAX_LOCK (currently set to 10) times in succession, it raises a SCAN_ALARM.
PACT	Processing Active	See Application Developers Guide for details on usage. PACT is TRUE while the record is being processed. For asynchronous records PACT can be TRUE from the time record processing is started until the asynchronous completion occurs. As long as PACT is TRUE, dbProcess will not call the record processing routine.
FLNK	Forward Link	This field is a database link. If FLNK is specified, processing this record will force a processing of the scan passive forward link record.

3. Alarm Fields

These fields indicate the status and severity of alarms, or else determine the how and when alarms are triggered. For a further explanation of database alarms, see *Alarm Specification, Chapter 1, 4*. Of course, many records have alarm-related fields not common to all records. These fields are listed and explained in the appropriate chapter on each record.

3.1. Field Summary

Field	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
STAT	GBLCHOICE	No	UDF_ALARM	Yes	No	Yes	No
SEVR	GBLCHOICE	No	INVALID_ALARM	Yes	No	Yes	No
NSTA	GBLCHOICE	No	0	Yes	No	No	No
NSEV	GBLCHOICE	No	0	Yes	No	No	No
ACKS	GBLCHOICE	No	0	Yes	No	No	
ACKT	GBLCHOICE	No	11	Yes	No	No	
UDF	UCHAR	No	1	Yes	Yes	No	Yes

3.2. Field Description

Name	Summary	Description
STAT	Current Alarm Status	These four fields are the alarm status and severity fields. STAT and SEVR are the values seen outside database access. NSTA and NSEV are the fields the database access, record support, and device support use to set new alarm status and severity values. Whenever any software component discovers an alarm condition, it uses the following macro function: recGblSetSevr(precord,new_status,new_severity) This ensures that the current alarm severity is set equal to the highest outstanding alarm. The file alarm.h defines all allowed alarm status and severity values.
SEVR	Current Alarm Severity	
NSTA	New Alarm Status	
NSEV	New Alarm Severity	
ACKS	Alarm Acknowledge Severity	Highest severity unacknowledged alarm
ACKT	Alarm Acknowledge Transient	Is it necessary to acknowledge transient alarms?
UDF	VAL Undefined	UDF is initialized to TRUE at IOC initialization. Record and device support routines which write to the VAL field are responsible for setting UDF to FALSE.

4. Device Fields

These fields contain information about the device and record support used by a record.

4.1. Field Summary

Field	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SPVT	NOACCESS	No	4	No	No	No	
RSET	NOACCESS	No	4	No	No	No	
DSET	NOACCESS	No	4	No	No	No	
DPVT	NOACCESS	No	4	No	No	No	

4.2. Field Description

Name	Summary	Description
SPVT	Scan Private	This field is for private use of the scanning system.
RSET	Address of Record Support Entry Table	See Application Developers Guide for details on usage.
DSET	Address of Device Support Entry Table	This address of the device support entry table for this record. The value of this field is determined at IOC initialization time. Record support routines use this field to locate their device support routines.
DPVT	Device Private	This field is for private use of the device support modules.

5. Debugging Fields

These fields can aid in the debugging process.

5.1. Field Summary

Field	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
TPRO	UCHAR	No	0	Yes	Yes	No	No
BKPT	NOACCESS	No	1	No	No	No	

5.2. Field Description

Name	Summary	Description
TPRO	Trace Processing	If this field is set TRUE, a message is printed each time this record is processed and a message is printed for each record processed as a result of this record being processed
BKPT	BreakPoint	Holds a pointer to the breakpoint table specified in LINR, if any.

6. Miscellaneous Fields

These are miscellaneous fields common to all record types.

6.1. Field Description

Field	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NAME	STRING [29]	Yes	0	Yes	No	No	
DESC	STRING [29]	Yes	Null	Yes	Yes	No	No
ASG	STRING [29]	Yes	Null	Yes	Yes	No	
TSE	SHORT	No	0	Yes	Yes	No	
TSEL	INLINK	Yes	Null	No	No	N/A	No
DTYP	DEVCHOICE	Yes	0	Yes	No	No	
MLOK	NOACCESS	No	8	No	No	No	
MLIS	NOACCESS	No	12	No	No	No	
DISP	UCHAR	No	0	Yes	Yes	Yes	No
PUTF	UCHAR	No	0	Yes	No	No	
RPRO	UCHAR	No	0	Yes	No	No	
ASP	NOACCESS	No	4	No	No	No	
PPN	NOACCESS	No	4	No	No	No	
PPNN	NOACCESS	No	4	No	No	No	
RDES	NO_ACCESS						
TIME	NOACCESS	No	8	Option	No	No	No

6.2. Field Description

Name	Summary	Description
NAME	Record Name	An arbitrary 28 character record name supplied by the application developer. This name is the means of identifying a specific record. It must have a unique value across all IOCs attached to the same local area subnet.

Name	Summary	Description
DESC	Description	An arbitrary 28 character record description supplied by the application developer.
ASG	Access Security Group	A character string value defining the access security group for this record. If left NULL, the record is placed in group DEFAULT.
TSE	Time Stamp Event	The event number for time stamp. This is only meaningful if the IOC has an associated hardware event receiver. See ‘er' record for details.
TSEL	Time Stamp Event Link	An input link for obtaining the time stamp event number.
DTYP	Device Type	This field specifies the device type for the record. Each record type has its own set of device support routines which are specified in devSup.ASCII. If a record type does not have any associated device support, DTYP and DSET are meaningless.
MLOK	Monitor Lock	The lock used by the monitor routines when the monitor list is being used. The list is locked whenever monitors are being scheduled, invoked, or when monitors are being added to or removed from the list. This field is accessed only by the dbEvent routines.
MLIS	Monitor List	This is the head of the list of monitors connected to this record. Each record support module is responsible for triggering monitors for any fields that change as a result of record processing. Monitors are present if mlis count is greater than zero. The call to trigger monitors is: db_post_event(precord,&data,mask), where “mask” is some combination of DBE_ALARM, DBE_VALUE, and DBE_LOG.
DISP	Disable putFields	If this field is set to TRUE, then all dbPutFields (normally issued by channel access) directed to this record are ignored except to the field DISP itself.
PUTF	dbPutField Process	Did dbPutField cause the current record processing?
RPRO	Reprocess	Reprocess record when current processing completes.
ASP	Access Security Private	
PPN	Address of putNotify	Address of putNotify callback.
PPNN	Next Record for putNotify	Next record for PutNotify.
RDES	Address of dbRecordType	
TIME	Time	The time when this record was last processed, in standard format.

Chapter 3: *Fields Common to Many Record Types*

1. Introduction

This chapter describes input and output fields that are common to multiple record types. These fields have the same meaning whenever they are used.

2. Input Records

2.1. Common Fields

Name	Summary	Description
INP	Input Link	This field is used by the device support routines to obtain input. For soft analog records it can be a constant, a database link, or a channel access link.
DTYP	Device Type	DTYP specifies the name of the device support module that will input values. Each record type has its own set of device support routines. If a record type does not have any associated device support, DTYP is meaningless.
RVAL	Raw Value	Whenever possible this field contains the raw data value exactly as it is obtained from the hardware or from the associated device driver and before it undergoes any conversions. The `Soft Channel` device support module reads values directly into VAL, bypassing this field.
VAL	Value	This is the record's final value, after any needed conversions have been performed.
SIMM	Simulation Mode	This field has either the value YES or NO. By setting this field to YES, the record can be switched into simulation mode of operation. While in simulation mode, input will be obtained from SIOL instead of INP.

Name	Summary	Description
SIML	Simulation Mode Location	This field can be a constant, a database link, or a channel access link. If SIML is a database or channel access link, then SIMM is read from SIML. If SIML is a constant link then SIMM is initialized with the constant value but can be changed via dbPuts.
SVAL	Simulation Value	This is the record's input value, in engineering units, when the record is switched into simulation mode, i.e. when SIMM is set to YES.
SIOL	Simulation Value Location	This field can be a constant, a database link, or a channel access link. If SIOL is a database or channel access link, then SVAL is read from SIOL. If SIOL is a constant link then SVAL is initialized with the constant value but can be changed via dbPuts.
SIMS	Simulation Mode Alarm Severity	When this record is in simulation mode, it will be put into alarm with this severity and a status of SIMM.

2.2. Device Input

A device input routine normally returns one of the following values to its associated record support routine:

- 0: Success and convert. The input value is in RVAL. The record support module is expected to compute VAL from RVAL.
- 2: Success, but don't convert. The device support module can specify this value if it does not want any conversions. It might do this for two reasons:
- A hardware error is detected (in this case, it should also raise an alarm condition).
- The device support routine reads values directly into the VAL field and then sets UDF to FALSE.

2.3. Soft Input

In almost all cases, two special device support modules are provided: `Soft Channel` and `Raw Soft Channel`. Both allow INP to be a constant, a database link, or a channel access link. The `Soft Channel` device support module reads input directly into the VAL field and specifies that no conversion of any type should be performed. Thus, it allows the record to hold values corresponding to the C data type of the VAL field. Note that for soft input, RVAL is not used. The `Raw Soft Channel` support module reads input into RVAL and asks that any specified conversions be performed.

The device support read routine normally calls `recGblGetLinkValue()` which performs the following steps:

- If the INP link type is CONSTANT `recGblGetLinkValue()` does nothing and returns with a status of zero.
- If the INP link type is DB_LINK, then `dbGetLink()` is called to obtain a new input value. If `dbGetLink` returns an error, a LINK_ALARM with a severity of INVALID_ALARM is raised. `RecGblGetLinkValue()` returns the status returned by `dbGetLink()`.
- If the INP link type is CA_LINK, then `dbCaGetLink()` is called to obtain a new input value. If `dbCaGetLink()` returns an error, a LINK alarm with a severity of INVALID is raised. `RecGblGetLinkValue()` returns the status of `dbCaGetLink()`.

If the return status of `recGblGetLinkValue()` is zero and the INP link type is not CONSTANT, then UDF is set to FALSE. The device support read routine normally returns the status of recGblGetLinkValue.

2.4. Simulation Mode

A record can be switched into simulation mode of operation by setting the value of SIMM to YES. During simulation, the record will be put into alarm with a severity of SIMS and a status of SIMM_ALARM. While in simulation mode, input values, in engineering units, will be obtained from SIOL instead of INP. Also, while the record is in simulation mode, there will be no raw value conversion and no calls to device support when the record is processed.

Normally input records contain a private `readValue()` routine which performs the following steps:

- If PACT is TRUE, the device support read routine is called, status is set to its return code, and readValue returns.
- Call `recGblGetLinkValue()` to get a new value for SIMM if SIML is a DB_LINK or a CA_LINK.
- Check value of SIMM.
- If SIMM is NO, then call the device support read routine, set status to its return code, and return.
- If SIMM is YES, then call recGblGetLinkValue to read the input value from SIOL into SVAL. If success, then set VAL to SVAL and UDF to FALSE and set status to 2 (don't convert) if input record supports conversion. If SIMS is greater than zero, set alarm status to SIMM and severity to SIMS. Set status to the return code from recGblGetLinkValue and return.
- IF SIMM is not YES or NO, a SOFT alarm with a severity of INVALID is raised, and return status is set to -1.

3. Output Records

3.1. Common Fields

Name	Summary	Description
OUT	Output Link	This field is used by the device support routines to decide where to send output. For soft records, it can be a constant, a database link, or a channel access link. If the link is a constant, the result is no output.
DTYP	Device Type	DTYP specifies the name of the device support module that will receive values. Each record type has its own set of device support routines. If a record type does not have any associated device support, DTYP is meaningless.
VAL	Value	This is the desired value before any conversions to raw output have been performed.
OVAL	Output Value	OVAL is used to decide when to invoke monitors. Archive and value change monitors are invoked if OVAL is not equal to VAL. If a record type needs to make adjustments, OVAL is used to enforce the maximum rate of change limit before converting the desired value to a raw value.
RVAL	Raw Value	Whenever possible this is the actual value sent to the hardware itself or to the associated device driver.
RBV	Read Back Value	Whenever possible this is the actual read back value obtained from the hardware itself or from the associated device driver.
DOL	Desired Output Location (an Input Link)	DOL can be a constant, a database link, or a channel access link. There is no device support associated with DOL. If DOL is a database or channel access link and OMSL is CLOSED_LOOP, then VAL is obtained from DOL.
OMSL	Output Mode Select	This field has either the value SUPERVISORY or CLOSED_LOOP. DOL is used to determine VAL only if OMSL has the value CLOSED_LOOP. By setting this field the record can be switched between supervisory and closed loop mode of operation. While in closed loop mode, the VAL field cannot be set via dbPuts.
OIF	Output Full or Incremental (analog output record only)	This field, which is only used when input is obtained from DOL, determines if the value obtained from DOL is an increment to add to the current VAL or is the actual VAL desired.
SIMM	Simulation Mode	This field has either the value YES or NO. By setting this field to YES, the record can be switched into simulation mode of operation. While in simulation mode, output will be written to SIOL instead of OUT.
SIML	Simulation Mode Location	This field can be a constant, a database link, or a channel access link. If SIML is a database or channel access link, then SIMM is read from SIML. If SIML is a constant link then SIMM is initialized with the constant value but can be changed via dbPuts.

Name	Summary	Description
SIOL	Simulation Value Location	This field can be a constant, a database link, or a channel access link. If SIOL is a database or channel access link, then the output value is written to SIOL. If this link is a constant, the result is no output.
SIMS	Simulation Mode Alarm Severity	When this record is in simulation mode, it will be put into alarm with this severity and a status of SIMM_ALARM.
IVOA	Invalid Alarm Output Action	Whenever the record is put into INVALID alarm severity IVOA specifies an action. IVOA can be one of the following actions: Continue normally, Don't drive outputs, Set output equal to IVOV
IVOV	Invalid Alarm Output Value, In Engineering Units	When new severity has been set to INVALID alarm and IVOA is “Set output equal to IVOV”, then, VAL is set to IVOV and converted to RVAL before device support is called.

3.2. Soft Output

Normally two soft output device support modules are provided Soft and Raw Soft. Both allow the output link OUT to be a constant, a database link, or a channel access link. It is normally meaningless to use constant output links. The Soft support module writes output from the value associated with OVAL or VAL (if OVAL does not exist). The `Raw Soft Channel` support module writes the value associated with the RVAL field after conversion has been performed.

The device support write routine normally calls recGblPutLinkValue which performs the following steps:

- If the OUT link type is CONSTANT recGblPutLinkValue does nothing and returns with a status of zero.
- If the OUT link type is DB_LINK, then dbPutLink is called to write the current value. If dbPutLink returns an error, a LINK_ALARM with a severity of INVALID_ALARM is raised. RecGblPutLinkValue returns the status of dbPutLink.
- If the OUT link type is CA_LINK, then dbCaPutLink is called to write the current value. If dbCaPutLink returns an error, a LINK_ALARM with a severity of INVALID_ALARM is raised. RecGblPutLinkValue returns the status of dbCaPutLink.

The device support write routine normally returns the status of recGblPutLinkValue.

3.3. Output Mode Select

The fields DOL and OMSL are used to allow the output record to be part of a closed loop control algorithm. OMSL is meaningful only if DOL refers to a database or channel access link. It can have the values SUPERVISORY or CLOSED_LOOP. If the mode is SUPERVISORY, then nothing is done to VAL. If the mode is CLOSED_LOOP and the record type does not contain an OIF field, then each time the record is processed, VAL is set equal to the value obtained from the location referenced by DOL. If the mode is CLOSED_LOOP in record types with an OIF field and OIF is Full, VAL is set equal to the value obtained from the location referenced by DOL; if OIF is Incremental VAL is incremented by the value obtained from DOL.

3.4. Simulation Mode

An output record can be switched into simulation mode of operation by setting the value of SIMM to YES. During simulation, the record will be put into alarm with a severity of SIMS and a status of SIMM_ALARM. While in simulation mode, output values, in engineering units, will be written to SIOL instead of OUT. However, the output values are never converted.

```
        — (SIMM = FALSE?) INP -> RVAL —(maybe convert) -> VAL
     /
SIML -> SIMM
     \
        — (SIMM = TRUE?) SIOL -> SVAL —(never convert) -> RVAL
```

Also, while the record is in simulation mode, there will be no calls to device support during record processing.

Normally output records contain a private `writeValue()` routine which performs the following steps:

- If PACT is TRUE, the device support write routine is called, status is set to its return code, and readValue returns.
- Call recGblGetLinkValue to get a new value for SIMM if SIML is a DB_LINK or a CA_LINK.
- Check value of SIMM.
- If SIMM is NO, then call the device support write routine, set status to its return code, and return.
- If SIMM is YES, then call recGblPutLinkValue to write the output value from VAL or OVAL to SIOL. Set alarm status to SIMM and severity to SIMS, if SIMS is greater than zero. Set status to the return code from recGblPutLinkValue and return.
- If SIMM not one of the above, a SOFT alarm with a severity of INVALID is raised, and return status is set to -1.

3.5. Invalid Alarm Output Action

Whenever an output record is put into INVALID alarm severity, IVOA specifies an action to take. The record support process routine for each output record contains code which performs the following steps.

- If new severity is less than INVALID, then call writeValue:
- Else do the following:
- If IVOA is CONTINUE, then call writeValue.
- If IVOA is NO_OUTPUT, then do not write output.
- If IVOA is OUTPUT_IVOV, then set VAL to IVOV, call convert if necessary, and then call writeValue.
- If IVOA not one of the above, an error message is generated.

Chapter 4: ai—Analog Input

1. Introduction

The normal use for this record type is to obtain an analog value from hardware and then convert it to engineering units. Most device support modules obtain values from hardware. The record supports alarm limits, conversion to engineering units, smoothing, and graphics and control limits.

Two soft device modules—`Soft Channel` and `Raw Soft Channel`—are provided to obtain input via database or channel access links or via `dbPutField` or `dbPutLink` requests. The first module reads values directly into VAL. The second reads values into RVAL. These values are then converted just like raw values obtained from hardware device support modules. If soft device support with a constant INP link is chosen, then the VAL field can be modified via dbPuts.

The fields fall into the following groups of parameters:

1. scan parameters
2. read and convert parameters
3. operator display parameters
4. alarm parameters
5. monitor parameters
6. run-time parameters

2. Scanning Parameters

The analog input record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Read and Convert Parameters

These parameters determine where the record gets its input and how it converts the raw signal to engineering units. For records that obtain input from devices or that use the `Raw Soft Channel` device support, the device support routines will return the value of this device into the RVAL field. Unless the LINR conversion field specifies NO CONVERSION, the proper conversion algorithm will be performed and the resulting value will be placed in the VAL field.

A further explanation of these fields follows.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Value Field	DOUBLE	No	0	Yes	Yes	Yes	Yes
INP	Input Link	INLINK	Yes	0	No	No	N/A	No
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	
LINR	Type of Conversion	CVTCHOICE	Yes	0	Yes	Yes	No	Yes
RVAL	Raw Value	LONG	No	0	Yes	Yes	Yes	Yes
ROFF	Raw Value Offset	LONG	No	0	Yes	Yes	No	Yes
EGUF	Engineering Units Full	FLOAT	Yes	0	Yes	Yes	No	Yes
EGUL	Engineering Units Low	FLOAT	Yes	0	Yes	Yes	No	Yes
AOFF	Adjustment Offset	FLOAT	Yes	0	Yes	Yes	No	Yes
ASLO	Adjustment Slope	FLOAT	Yes	1	Yes	Yes	No	Yes
ESLO	Slope for Linear Conversions	DOUBLE	No	1	Yes	No	No	No
SMOO	Smoothing Factor	FLOAT	Yes	0	Yes	Yes	No	No

3.1. Input Specification

As stated in the introduction to this chapter, what is entered in the INP field of the analog input record determines its run-time behavior. For an analog record that obtains its value from hardware, the address of the I/O card must appear in the INP field, and the name of the device support module must appear in the device type field (DTYP). See *Address Specification, Chapter 1, 2*, for information on the format of hardware addresses. Be aware that the format differs between types of cards. You can see a list of the device support modules currently supported at the user's local site by using the `dbst` utility in R3.13.

The INP field can also specify a constant value, a channel access link, or a database link. When configured with a constant value, the record's VAL field is initialized with the value given to the field and the VAL field can be changed using dbPuts. See *Address Specification, Chapter 1, 2*, for information on database and channel access links.

When INP is a constant, a database link, or a channel access link either one of two soft device support modules, `Raw Soft Channel` or `Soft Channel`, must be specified in DTYP field. See Section 10.3, *Device Support For Soft Records*, in this chapter for more on how these device support modules work.

3.2. Conversion Related Fields

The LINR field determines if a conversion is performed and which conversion algorithm is used to convert the raw value (RVAL). No conversions are performed if the Soft Channel device support module is used. For other records that use other device support modules, the LINR field determines what adjustments or conversions are made on the raw value. The LINR field specifies either `LINEAR`, `NO CONVERSION`, or the name of a breakpoint table, such as `typeKdegC`. LINEAR specifies a linear conversion; NO CONVERSION, no conversion at all; and a breakpoint table, a breakpoint conversion. Note that the EGUF, EGUL, and ESLO fields are not used when a breakpoint table or NO CONVERSION is specified.

ROFF	Computed by the device support routines It is used by all records except those that use `Soft Channel`.
EGUF EGUL	The user must calculate these fields when configuring the database. They are used to calculate the value of ESLO. See *Conversion Specification, Chapter 1, 3*, for more information on how to calculate these fields. The are used only if LINR specifies LINEAR.
AOFF ASLO	Configured by the user, and used for all records but `Soft Channel` records.
ESLO	Computed by device support using EGUF and EGUL. Used if LINR specifies `LINEAR`.
SMOO	A value between 1 and 0 specified by the user, with 0 meaning no smoothing and 1 meaning ultimate smoothing (in fact, the data value will never change). See *Conversion Specification, Chapter 1, 3*, for more information on what this field does. It is used for all records but `Soft Channel` records.

The record processing routine performs the following algorithm for all records except those that use the `Soft Channel` device support routine. Be aware that step 3 is performed only when the record specifies LINEAR:

1. Val= RVAL + ROFF
2. Val = Val * ASLO + AOFF

If the conversion algorithm is `LINEAR`, the raw value is converted via the equation:

3. val = val * ESLO + EGUL

If the conversion is via a breakpoint table, the new value is obtained.

The next step is to apply the following smoothing algorithm:

4. if SMOO is 0 or INIT is True, VAL = val
5. else VAL = val * (1 - SMOO) + Previous_value * SMOO

Since VAL is now defined, the last step is to set UDF to FALSE.

For a complete explanation on conversion parameters, see *Conversion Specification, Chapter 1, 3*. To see how `Raw Soft Channel` device support uses these parameters, see Section 10.3, *Device Support For Soft Records*, in this chapter.

4. Operator Display Parameters

These parameters are used to present meaningful data to the operator. They display the value and other parameters of the analog input either textually or graphically. EGU is a string of up to 16 characters describing the units that the analog input measures. It is retrieved by the `get_units` record support routine.

The HOPR and LOPR fields set the upper and lower display limits for the VAL, HIHI, HIGH, LOW, and LOLO fields. Both the `get_graphic_double` and `get_control_double` record support routines retrieve these fields.

The PREC field determines the floating point precision with which to display VAL. It is used whenever the `get_precision` record support routine is called.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	null	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

5. Alarm Parameters

The possible alarm state for analog inputs are the SCAN, READ, and limit alarms. The SCAN and READ alarms are called by the record or device support routines.

The limit alarms are configured by the user in the HIHI, LOLO, HIGH, and LOW fields using numerical values. For each of these fields, there is a corresponding severity field which can be either NO_ALARM, MINOR, or MAJOR. See *Alarm Specification, Chapter 1, 4*, for a complete explanation of alarms and these fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HIHI	Hihi Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOLO	Lolo Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HIGH	High Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOW	Low Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HHSV	Hihi Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LLSV	Lolo Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HSV	High Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LSV	Low Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HYST	Alarm Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

6. Monitor Parameters

These parameters are used to determine when to send monitors placed on the VAL field. The monitors are sent when the value field exceeds the last monitored field by the appropriate deadband. If these fields have a value of zero, everytime the value changes, a monitor will be triggered; if they have a value of -1, everytime the record is scanned, monitors are triggered. The ADEL field is used by archive monitors and the MDEL field for all other types of monitors. See *Monitor Specification, Chapter 1, 5*, for a complete explanation of monitors.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ADEL	Archive Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
MDEL	Monitor, i.e. value change, Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

7. Run-Time Parameters and Simulation Mode Parameters

These parameters are used by the run-time code for processing the analog input. They are not configurable by the user, but many can be modified after initialization. They represent the current state of the analog input. Many of them are used to process the analog input more efficiently.

The ORAW field is used to decide if monitors should be triggered for RVAL when monitors are triggered for VAL. The LALM, MLST, and ALST fields are used to implement the hysteresis factors for monitors on the VAL field.

The PBRK field contains a pointer to the breakpoint table specified in the LINR field (if any). The LBRK field indicates the name of the last breakpoint table used (if any).

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ORAW	Old Raw Value	LONG	No	0	Yes	No	No	No
LALM	Last Alarm Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No
ALST	Last Archiver Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
MLST	Last Value Change Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No
INIT	Initialize	SHORT	No	0	Yes	No	No	No
PBRK	Address of Breakpoint Table	NOACCESS	No	4	No	No	No	
LBRK	Last Breakpoint	SHORT	No	0	Yes	No	No	No

The following fields are used to operate the analog input in the simulation mode. See Chapter 3, *Fields Common to Many Record Types*, for more information on these fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No
SVAL	Simulation Value	DOUBLE	No	0	Yes	Yes	No	No
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

8. Record Support Routines

The following are the record support routines that would be of interest to an application developer. Other routines are the `get_units`, `get_precision`, `get_graphic_double`, and `get_control_double`, all of which are used for the monitor parameters.

init_record

This routine initializes SIMM with the value of SIML if SIML type is CONSTANT link or creates a channel access link if SIML type is PV_LINK. SVAL is likewise initialized if SIOL is CONSTANT or PV_LINK.

This routine next checks to see that device support is available and a device support `read_ai()` routine is defined. If either does not exist, an error message is issued and processing is terminated.

INIT is then set to TRUE.

If device support includes `init_record`, it is called.

process

See next section.

special

The only special processing for analog input records is SPC_LINCONV. which is invoked whenever any of the fields LINR, EGUF, EGUL or ROFF is changed.

If the device support routine special_linconv exists, it is called.

INIT is set TRUE. This causes PBRK, LBRK, and smoothing to be re-initialized.

get_value

Fills in the values of the structure valueDes so that they refer to VAL.

get_alarm_double

Sets the following values:

upper_alarm_limit = HIHI

upper_warning_limit = HIGH

lower_warning_limit = LOW

lower_alarm_limit = LOLO

9. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. readValue is called. See *Output Records, Chapter 3, 3*, for details.
3. If PACT has been changed to TRUE, the device support read routine has started but has not completed writing the new value. In this case, the processing routine merely returns, leaving PACT TRUE.
4. PACT is then set to TRUE, TIME is set to tslocaltime and the return status value of readValue is checked. convert is called only if status is 0. If status is 2, then convert is not called, but status is reset to 0.
5. Perform conversion if necessary: After conversions (if any), UDF is set to FALSE.
6. Check alarms: This routine checks to see if the new VAL causes the alarm status and severity to change. If so, NSEV, NSTA and LALM are set. It also honors the alarm hysteresis factor (HYST). Thus the value must change by more than HYST before the alarm status and severity is lowered.
7. Check to see if monitors should be invoked:
    - •Alarm monitors are invoked if the alarm status or severity has changed.
    - •Archive and value change monitors are invoked if ADEL and MDEL conditions are met.
    - •Monitors for RVAL are checked whenever other monitors are invoked.
    - •NSEV and NSTA are reset to 0.

8. Scan forward link if necessary, set PACT and INIT to FALSE, and return.

10. Device Support

10.1. Fields Of Interest To Device Support

Each analog input record must have an associated set of device support routines. The primary responsibility of the device support routines is to obtain a new raw analog input value whenever read_ai is called. The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See Chapter 2, *Fields Common to All Record Types* for an explanation of these fields.
DPVT	Device Private	
UDF	VAL Undefined	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
VAL	Value	This field is used by device support only if it obtains a value already converted to engineering units. See RVAL below.
INP	Input Link	This field is used by the device support routines to locate its input.
EGUF	Engineering Units Full	These fields are used to calculate ESLO. Note that these fields correspond to the high and low hardware limits.
EGUL	Engineering Unit Low	
ESLO	Slope	These fields are used for linear conversions from raw to engineering units. The device support routines must calculate these fields unless they obtain values already in engineering units.
ROFF	Raw Offset	
RVAL	Raw Value	It is the responsibility of the device support routine to give this field a value. If the device support routine obtains a value already in engineering units, it should place the value in VAL and return a value of 2.

10.2. Device Support Routines

Device support consists of the following routines:

report

```
report (FILE fp, paddr)
```

Not currently used.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record (precord)
```

This routine is optional. If provided, it is called by the record support init_record routine.

get_ioint_info

```
get_ioint_info (int cmd,struct dbCommon *precord,IOSCANPVT
    *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

read_ai

```
read_ai(precord)
```

This routine must provide a new input value. Asynchronous device support routines will return with PACT set to TRUE. If PACT is TRUE, the process routine will just return and not continue processing. When the asynchronous routine completes, it can call process which will again call read_ai.

Because PACT is still TRUE read_ai knows that this is a request to retrieve the data obtained by the previous call. When finished, read_ai should set PACT to FALSE and return one the following values:

- 0: Success. A new raw value is placed in RVAL. convert will be called.
- 2: Success, but don't call convert. This is useful if read_ai obtains a value already converted to engineering units or in the event a hardware failure is detected.
- Other: Error.

special_linconv

```
special_linconv(precord,after)
```

This routine is called whenever any of the fields LINR, EGUF, EGUL or ROFF is modified.

10.3. Device Support For Soft Records

Two soft device support modules `Soft Channel` and `Raw Soft Channel` are provided for input records not related to actual hardware devices. The INP link type must be either CONSTANT, DB_LINK or CA_LINK.

Soft Channel

This module places a value directly in VAL. read_ai always returns a value of 2, which means that no conversion will ever be attempted.

If the INP link type is constant, then the constant value is stored into VAL by init_record, and UDF is set to FALSE. If the INP link type is PV_LINK, then dbCaAddInlink is called by init_record.

read_ai calls recGblGetLinkValue to read the current value of VAL. See *Soft Input, Chapter 3, 2.3*, for details.

If the return status of recGblGetLinkValue is zero, then read_ai sets UDF to FALSE. The status of recGblGetLinkValue is returned.

If soft support is chosen, the following fields become meaningless: LINR, EGUF, EGUL, ESLO, ROFF, AOFF, ASLO, and SMOO. The read_ai routine always returns a value of 2 which means don't convert.

Raw Soft Channel

This module is like the previous except that it places its value in RVAL and read_ai returns a value of 0. Thus the record processing routine will convert the raw value in the normal way.

If raw soft support is chosen, the fields EGUF and EGUL become meaningless. ESLO and ROFF always have their default values of 1 and 0.

Chapter 5: ao - Analog Output

1. Introduction

The normal use for this record type is to output values to digital-analog converters. It is used for all analog outputs to hardware. It can also be used to write floating point values to other records via database or channel access links. How the user configures the output link determines if the record sends its value to a hardware device, a channel access link, or a database link. The desired output can be controlled by either an operator or a state program, or it can be fetched from another record.

The record supports alarm limits, conversion from/to engineering units, and graphics and control limits. The analog output fields fall into the following categories:

scan parameters

desired output parameters

convert and write parameters

operator display parameters

alarm parameters

monitor parameters

run-time parameters.

2. Scan Parameters

The analog output record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Desired Output Parameters

The analog output record must specify where the desired output to be written should originate. The desired output should be in engineering units. The first field that determines where the desired output originates is the output mode select field (OMSL), which can has two choices—`closed_loop` or `supervisory`. If `supervisory` is specified, the value in the VAL field can be set externally via dbPuts at run-time. If `closed_loop` is specified, the VAL field's value is obtained from the address specified in the desired output location field (DOL), which can be either a database link or a channel access link. To achieve continuous control, a database link to a control algorithm record should be entered in the DOL field.

When VAL is obtained from DOL, the OIF field decides whether the value obtained from DOL is an increment to be added to the current VAL or if the value obtained from DOL is the actual value. The OIF field has two choices, `Incremental` or `Full`. The OIF and OMSL fields, in addition to being configurable, can also be changed during run-time. (OIF is not used when OMSL is set to SUPERVISORY.)

The VAL field's value is forced to be within the limits specified in the fields DRVH and DRVL, which are configured by the designer:

```
DRVL<=VAL<=DRVH
```

Note: If nothing is entered for DRVH and DRVL, the output value will never change.

The VAL field is then adjusted by the OROC field's value if the OROC field is not zero. The OROC field determines the maximum change in value that occurs each time the record is processed. The value adjusted by OROC is then used by OVAL. If the address contained in the output link (see next section) is a channel access or database link and if `Soft Channel` device support is specified, the value in OVAL is sent to the address in the OUT field. Otherwise, a conversion process is performed. The next section on convert and write parameters explains how this value is converted before being written.

See *Address Specification, Chapter 1, 2*, for information on specifying links. *Scanning Specification, Chapter 1, 1*, explains the effect of database linkage on scanning.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OMSL	Output Mode Select	GBLCHOICE	Yes	0	Yes	Yes	No	No
DOL	Desired Output Location (an Input Link)	INLINK	Yes	0	No	No	N/A	No
OIF	Out Full or Incremental	RECCHOICE	Yes	0	Yes	Yes	No	No
DRVH	Drive High	FLOAT	Yes	0	Yes	Yes	No	Yes
DRVL	Drive Low	FLOAT	Yes	0	Yes	Yes	No	Yes
VAL	Value	DOUBLE	No	0	Yes	Yes	Yes	Yes
OROC	Maximum Output Rate of Change	FLOAT	Yes	0	Yes	Yes	No	No
OVAL	Output Value	DOUBLE	No	0	Yes	Yes	Yes	No

4. Convert and Write Parameters

For analog output records that do not use the `Soft Channel` device support routine, the specified conversions (if any) are performed on the OVAL field and the resulting value in the RVAL field is sent to the address contained in the output link after it is adjusted by the values in the AOFF and ASLO fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OUT	Output Link	OUTLINK	Yes	0	No	No	N/A	No
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	
LINR	Type of Conversion	CVTCHOICE	Yes	0	Yes	Yes	No	Yes
RVAL	Raw Value	LONG	No	0	Yes	Yes	Yes	Yes
ROFF	Raw Value Offset	LONG	No	0	Yes	Yes	No	Yes
EGUF	Engineering Units Full	FLOAT	Yes	0	Yes	Yes	No	Yes
EGUL	Engineering Units Low	FLOAT	Yes	0	Yes	Yes	No	Yes
AOFF	Adjustment Offset	FLOAT	Yes	0	Yes	Yes	No	Yes
ASLO	Adjustment Slope	FLOAT	Yes	1	Yes	Yes	No	Yes
ESLO	Slope for Linear Conversions	DOUBLE	No	1	Yes	No	No	No

4.1. Conversion Related Fields and the Conversion Process

Except for analog outputs that use `Soft Channel` device support, the LINR field determines if a conversion is performed and which conversion algorithm is used to convert OVAL to RVAL. The LINR field can specify LINEAR for linear conversions, NO CONVERSION for no conversions at all, or the name of a breakpoint table such as `typeKdegC` for breakpoint conversions.

Note that the ESLO, EGUF, and EGUL fields are only used for linear conversions. Also note that none of these fields have any significance for records that use the `Soft Channel` device support module.

ROFF	Computed by device support routines. Used to offset raw value. Used by all records but `Soft Channel` records.
EGUF EGUL	The user must calculate these fields when configuring the database. They are used to calculate the value ESLO, and are thus only significant for records that use linear conversions. See *Conversion Specification, Chapter 1, 3*, for more information on how to calculate these fields.
AOFF ASLO	These fields are adjustment parameters for the raw output values. They are applied to the raw output value after conversion from engineering units.

ESLO	Computed by device support using EGUF and EGUL. Used if LINR specifies LINEAR.

1. RVAL = (OVAL - EGUL) / ESLO + 0.5
2. RVAL = RVAL * ASLO + AOFF

If the conversion is via a breakpoint table, the new value is obtained.

3. RVAL = RVAL - ROFF

Since RVAL is now defined, the last step is to set UDF to FALSE.

To see how the `Raw Soft Channel` device support routine uses these fields, see Section 11.3, *Device Support For Soft Records*, in this chapter for more information.

4.2. Output Specification

The analog output record sends its desired output to the address in the OUT field. For analog outputs that write their values to devices, the OUT field must specify the address of the I/O card. In addition, the DTYP field must contain the name of the device support module. Be aware that the address format differs according to the I/O bus used. See *Address Specification, Chapter 1, 2*, for information on the format of hardware addresses. The user can see a list of the device support modules currently supported at the user's local site by using the `dbst` utility in R3.13.

For soft records the output link can be a database link, a channel access link, or a constant value. If the link is a constant, no output is sent. See *Address Specification, Chapter 1, 2*, for information on the format of database and channel access addresses.

5. Operator Display Parameters

These parameters are used to present meaningful data to the operator. They display the value and other parameters of the analog output either textually or graphically.

EGU is a string of up to 16 characters describing the units that the analog output measures. It is retrieved by the get_units record support routine.

The HOPR and LOPR fields set the upper and lower display limits for the VAL, OVAL, PVAL, HIHI, HIGH, LOW, and LOLO fields. Both the get_graphic_double and get_control_double record support routines retrieve these fields. If these values are defined, they must be in the range: DRVL<=LOPR<=HOPR<=DRVH.

The PREC field determines the floating point precision with which to display VAL, OVAL and PVAL. It is used whenever the get_precision record support routine is called.

See *Chapter 2, Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	null	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

The possible alarm conditions for analog outputs are the SCAN, READ, INVALID and limit alarms. The SCAN, READ, and INVALID alarms are called by the record or device support routines.

The limit alarms are configured by the user in the HIHI, LOLO, HIGH, and LOW fields, which must be floating-point values. For each of these fields, there is a corresponding severity field which can be either NO_ALARM, MINOR, or MAJOR.

See *Alarm Specification, Chapter 1, 4*, for a complete explanation of alarms and these fields. See *Chapter 3.5, Invalid Alarm Output Action*, for more information on the IVOA and IVOV fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HIHI	Hihi Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HIGH	High Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOW	Low Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOLO	Lolo Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HHSV	Hihi Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HSV	High Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LSV	Low Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LLSV	Lolo Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HYST	Alarm Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
IVOA	Invalid Alarm Output Action	GBLCHOICE	Yes	0	Yes	Yes	No	No
IVOV	Invalid Alarm Output Value, in eng. units	DOUBLE	Yes	0	Yes	Yes	No	No

7. Monitor Parameters

These parameters are used to specify deadbands for monitors on the VAL field. The monitors are sent when the value field exceeds the last monitored field by the specified deadband. If these fields have a value of zero, everytime the value changes, a monitor will be triggered; if they have a value of -1, everytime the record is processed, monitors are triggered. ADEL is the deadband for archive monitors, and MDEL the deadband for all other types of monitors. See *Monitor Specification, Chapter 1, 5*, for a complete explanation of monitors.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ADEL	Archive Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
MDEL	Monitor, i.e. value change, Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

8. Run-Time Parameters and Simulation Mode Parameters

These parameters are used by the run-time code for processing the analog output. They are not configurable. They represent the current state of the record. The record support routines use some of them for more efficient processing.

The ORAW field is used to decide if monitors should be triggered for RVAL when monitors are triggered for VAL. The RBV field is the actual read back value obtained from the hardware itself or from the associated device driver. It is the responsibility of the device support routine to give this field a value.

ORBV is used to decide if monitors should be triggered for RBV at the same time monitors are triggered for changes in VAL.

The LALM, MLST, and ALST fields are used to implement the hysteresis factors for monitor callbacks.

The INIT field is used to initialize the LBRK field and for smoothing.

The PBRK field contains a pointer to the current breakpoint table (if any), and LBRK contains a pointer to the last breakpoint table used.

The OMOD field indicates whether OVAL differs from VAL. It will be different if VAL or OVAL have changed since the last time the record was processed, or if VAL has been adjusted by OROC during the current processing...

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ORAW	Old Raw Value	LONG	No	0	Yes	No	No	No
RBV	Read Back Value	LONG	No	0	Yes	No	Yes	No
ORBV	Old read back value	LONG	No	0	Yes	No	No	No
LALM	Last Alarm Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No
ALST	Last Archiver Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
MLST	Last Value Change Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No
INIT	Initialize	SHORT	No	0	Yes	No	No	No
PBRK	Address of Breakpoint Table	NOACCESS	No	4	No	No	No	
LBRK	Last Breakpoint	SHORT	No	0	Yes	No	No	No
PVAL	Previous Data Value	DOUBLE	No	0	Yes	No	No	No
OMOD	OVAL modified?	LONG	No	Null	Yes	No	No	No

The following fields are used to operate the analog output in the simulation mode. See *Chapter 3, Fields Common to Many Record Types*, for more information on these fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No
SVAL	Simulation Value	DOUBLE	No	0	Yes	Yes	No	No
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

9. Record Support Routines

The following are the record support routines that would be of interest to an application developer. Other routines are the `get_units`, `get_precision`, `get_graphic_double`, and `get_control_double` routines.

init_record

This routine initializes SIMM if SIML is a constant or creates a channel access link if SIML is PV_LINK. If SIOL is PV_LINK a channel access link is created.

This routine next checks to see that device support is available. If DOL is a constant, then VAL is initialized with its value and UDF is set to FALSE.

The routine next checks to see if the device support write routine is defined. If either device support or the device support write routine does not exist, an error message is issued and processing is terminated.

If device support includes init_record, it is called.

INIT is set TRUE[. This causes PBRK, LBRK, and smoothing to be re-initialized If linear conversion is requested, then VAL is computed from RVAL using the algorithm:

```
VAL = (RVAL+ROFF) / ESLO + EGUL
```

and UDF is set to FALSE.

For breakpoint conversion, a call is made to cvtEngToRawBpt and UDF is then set to FALSE. PVAL is set to VAL.

process

See next section.

special

The only special processing for analog output records is SPC_LINCONV which is invoked whenever either of the fields LINR, EGUF, EGUL or ROFF is changed If the device support routine special_linconv exists it is called.

INIT is set TRUE. This causes PBRK, LBRK, and smoothing to be re-initialized.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_alarm_double

Sets the following values:

upper_alarm_limit = HIHI

upper_warning_limit = HIGH

lower_warning_limit = LOW

lower_alarm_limit = LOLO

10. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. Check PACT: If PACT is FALSE call fetch_values and convert which perform the following steps:
   - fetch_values:
     ```
     if DOL is DB_LINK and OMSL is CLOSED_LOOP get value   from DOL
     if OIF is INCREMENTAL then set value = value + VAL else value =
           VAL
     ```
   - convert:
     ```
     If Drive limits are defined force value to be within limits
     Set VAL equal to value
     Set UDF to FALSE.
     ```

```
If OVAL is undefined set it equal to value
If OROC is defined and not 0 make |value-OVAL| <=OROC
Set OVAL equal to value
Compute RVAL from OVAL. using linear or break point table
conversion. For linear conversions the algorithm is:
RVAL = (OVAL-EGUL)/ESLO -ROFF
```

•For break point table conversion a call is made to cvtEngToRawBpt.

3. Check alarms: This routine checks to see if the new VAL causes the alarm status and severity to change. If so, NSEV, NSTA and y are set. It also honors the alarm hysteresis factor (HYST). Thus the value must change by at least HYST before the alarm status and severity is reduced.
4. Check severity and write the new value. See *Invalid Alarm Output Action, Chapter 3, 3.5*, for details on how invalid alarms affect output records.
5. If PACT has been changed to TRUE, the device support write output routine has started but has not completed writing the new value. In this case, the processing routine merely returns, leaving PACT TRUE.
6. Check to see if monitors should be invoked:
   •Alarm monitors are invoked if the alarm status or severity has changed.
   •Archive and value change monitors are invoked if ADEL and MDEL conditions are met.
   •Monitors for RVAL and for RBV are checked whenever other monitors are invoked.
   •NSEV and NSTA are reset to 0.
7. Scan forward link if necessary, set PACT and INIT FALSE, and return.

11. Device Support

11.1. Fields Of Interest To Device Support

Each analog output record must have an associated set of device support routines. The primary responsibility of the device support routines is to output a new value whenever write_ao is called. The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See *Chapter 2, Fields Common to All Record Types* for an explanation of these fields.
DPVT	Device Private	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
OUT	Output Link	This field is used by the device support routines to locate its output.
EGUF	Engineering Units Full	These fields are used to calculate ESLO. Note that these fields correspond to the high and low hardware limits.
EGUL	Engineering Unit Low	

Name	Summary	Description
ESLO	Slope	These fields are used for linear conversions from raw to engineering units. The device support routines must calculate these fields unless they obtain values already in engineering units.
ROFF	Raw Offset	
RVAL	Raw Value	This is the value to write to OUT.

11.2. Device Support routines

Device support consists of the following routines:

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine. It returns a zero for success or a 2 for success, don't convert.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
    *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

write_ao

```
write_ao(precord)
```

This routine must output a new value. Asynchronous device support routines will return with PACT set to TRUE. If PACT is TRUE, the process routine will just return and not continue processing. When the asynchronous routine completes, it can call process which will again call write_ao. When finished, write_ao should set PACT to FALSE and return one the following values:

- 0: Success.
- other: Error.

special_linconv

```
special_linconv(precord,after)
```

This routine is called whenever either of the fields LINR, EGUF, EGUL or ROFF is modified.

11.3. Device Support For Soft Records

Two soft device support modules Soft Channel and Raw Soft Channel are provided for output records not related to actual hardware devices. The OUT link type must be either a CONSTANT, DB_LINK, or CA_LINK.

Soft Channel

This module writes the current value of OVAL.

If the OUT link type is PV_LINK, then dbCaAddInlink is called by init_record. init_record always returns a value of 2, which means that no conversion will ever be attempted.

write_ao calls recGblPutLinkValue to write the current value of VAL. See *Soft Output, Chapter 3, 3.2*, for details.

Raw Soft Channel

This module is like the previous except that it writes the current value of RVAL.

Chapter 6: The Archive Record

1. Introduction

The Archive record was developed to provide a way to store the values of a PV for use by an EPICS archiving client. A client that archives the states of one or more PVs typically uses Channel Access to establish monitors on a channel, or the client retrieves a PV's values directly using get requests, the client usually running on a different host than the host which is running the database. In the case that the network fails or the client or client's host crashes, then the Archive record provides a way to store values for the PV being archived within the IOC. In addition, the Archive record stores the time stamp for the value as well as the status and severity of the PV. The Archive record can then "flush" the stored values and time stamps out to the client the next time the record processes.

Each time the Archive record processes, it reads a value from the address specified by its input link (the INP field) and decides whether or not to store it in an array. Whether the Archive record stores the value depends on the configuration of the STIM, AVAR, RVAR, and PCAB fields. If the record decides to archive the value, the value will be written into the CVAL field, and then stored in an array of data structures, each structure having members for the value, time stamp, and status and severity. The value of the sample is also stored in another array. The seconds and nanoseconds parts of the time stamp for the value are also stored in separate arrays. Writing directly to the CVAL field will cause the record to process. The size of the arrays used by the Archive record is determined by the value of the NVAL field, which should be configured beforehand.

Each time the record is processed, it will determine if monitors should be posted for the Archive records arrays. That is, it will decide if the array of values and the time arrays should be flushed to the client. The Archive record flushes the arrays when the record is full, half-full, and when the number of seconds since the last flush exceeds the number of seconds specified in the FTIM field. FTIM should be configured before run-time.

scan parameters

read parameters

archive and monitor parameters

operator display parameters

run-time and simulation mode parameters

2. Scan Parameters

The Archive record has the standard fields for specifying under what circumstances it will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used.

3. Read Parameters

The INP field determines where the Archive record gets its input. It can be a database or channel access link, or a constant. Each time the record is processed, it reads the value from the specified address and writes it into the RVAL field. If constant, the RVAL field is initialized with the constant and can be changed via `dbPuts`. The Archive record does not support device input.

See *Address Specification, Chapter 1, 2*, for information on specifying links.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
RVAL	Raw Value	DOUBLE	No	0	Yes	Yes	No	No
INP	Input Link	INLINK	Yes	Null	Yes	No	N/A	No

4. Archive and Monitor Parameters

These configurable parameters are the core of the archive record and basically determine how it operates at run-time. The STIM, AVAR, RVAR, and PCAB fields determine whether or not the record will archive the value read into the RVAL field. If the time since the last value archived is greater than the number of seconds specified in the STIM field, the value read into RVAL will be archived. Otherwise, the record will then use the PCAB, and RVAR or AVAR fields to determine whether or not to save the value. The PCAB field is a menu-type field that has seven choices:

- `absolute`
- `relative`
- `abs && rel`
- `abs || rel`
- `on change`
- `allways`
- `never`

If the PCAB field specifies `absolute`, when the record reads in a value to RVAL it will archive the value only if the difference between RVAL and the last value archived is greater than the value specified in the AVAR field. The difference between the RVAL and the last archived value is calculated as an absolute value. If the PCAB field specifies `relative`, the record will archive the value in RVAL only if the difference between RVAL and the last archived value is greater than *n%* of the last value archived, where *n* is the value specified in the RVAR field. The difference between RVAL and the last archived value is calculated as an absolute value. Thus, if PCAB specifies `relative` and the RVAR specifies 10%, and if the last archived value was 100.0, then if the current value is less than 90 or greater than 110, then that means the value has changed by more than 10% and the value will be archived.

If the PCAB field specifies `abs && rel`, then the record will archive the value in RVAL only if both the above absolute and relative conditions are true. If the PCAB field specifies `abs || rel`, then the value will be archived if either condition holds true. If `on change` is specified, then the record will archive the value if it has changed from the last value archived; if `allways`, the record will archive the value in RVAL every time the record processes; and if `never`, the record will never archive the value.

If a value is to be archived, it is written into the CVAL field.

When the archive record archives a value, it archives the value, a time stamp for the value, as well as the status and severity of the PV's value. This information is collectively called a sample. The record keeps track of each sample with a data structure. An array of these data structures is allocated when the record is initialized. The size of this array is configured by the developer/user in the NVAL field. It's the responsibility of the configurer to make sure NVAL specifies an array of the appropriate size. In addition to an array to hold the samples, three more arrays are allocated of size NVAL: the first holds the value part of each sample; the second holds the time of each sample, where the time is the number of seconds past the epoch of the sample; the third holds the nanoseconds part of the sample's time stamp. Although these arrays are not directly accessible at run-time through read or write operations, the record posts the latter three arrays using monitors.

The record posts the above three arrays—an array with the sample's values, an array with the seconds part of the sample's time stamp, and an array with the nanoseconds part of the time stamp—when the difference between the current time and the last time the values were posted exceeds the number of seconds specified in the FTIM field. In addition, the arrays are posted when the arrays are full and half-full. Of course, these conditions are only checked when the record processes

The MASK field changes the above algorithm somewhat. It was intended to enable a special digital reading of any input. It's an unsigned integer field which, if non-zero, changes the algorithm of the record in the following way. Firstly, if the MASK field is set to true, the following conversion is performed on RVAL: its value is typecast to an unsigned integer, RVAL being read as floating-point number. Then this integer is ANDed (&) with the value in MASK. In deciding to archive this integer, if the interval specified in STIM has been exceeded, the value is archived, just as when MASK is zero. However, if STIM hasn't been exceed, if MASK is true, the PCAB field is ignored, and the record simply checks to see if the converted value is not equal to the last value archived. If it isn't, the value is archived.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
STIM	Save Time in Secnds	ULONG	Yes	900	Yes	No	No	No
PCAB	Percent or Absolute	RECCHOICE	Yes	0	Yes	Yes	No	No
AVAR	Absolute Difference	FLOAT	Yes	0	Yes	Yes	No	No
RVAR	Percent Difference	FLOAT	Yes	0	Yes	Yes	No	No
CVAL	Current Value	DOUBLE	No	0	Yes	Yes	No	Yes
NVAL	Number of Values	ULONG	Yes	100	Yes	No	No	No
FTIM	Flush Time in Sec	ULONG	Yes	900	Yes	No	No	No
MASK	Bit Mask	USHORT	Yes	0	Yes	No	No	No

5. Operator Display Parameters

These fields are used to give meaning to the record's data for the operator's sake. The HOPR and LOPR fields indicate the upper and lower limits for the CVAL and LVAL fields. In addition, the HOPR and LOPR fields determine the control limits for these fields. The PREC field determines the displayed precision for the CVAL, AVAR, and LVAL fields. The EGU field indicates the engineering units for the VAL, CVAL, AVAR, and LVAL fields.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NAME	Record Name	STRING [29]	Yes	Null	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Precision	SHORT	Yes	0	Yes	Yes	No	No
EGU	Engineering Units	STRING[16]	Yes	Null	Yes	Yes	No	No

6. Run-time Parameters

These parameters are all used for the record's internal processing or to access the record's data, except for the RES field which can be written to at run-time to reset the record. The ARCH, VALS, TIMS, and NSCS are all array fields of size NVAL. Their arrays contain the archived data. The ARCH field points to an array of data structures, each of which contains the value of the PV, the time stamp when the value was taken, and the status and severity of the PV when the value was taken. The VALS field points to an array containing only the value part of each sample; the TIMS field points to an array containing only the seconds part of the time stamp of each sample, while the NSCS field points to an array containing only the nanoseconds part of each sample. Each array is of the size specified in the NVAL field. None of these array fields are directly accessible at run-time. However, the VALS and TIMS arrays are accessible through the VAL and TIM fields.

The FLSH field is an internal flag that indicates whether or not the VALS, TIMS, and NSCS fields are to be flushed when monitors are posted.

When the user changes the RES field to a non-zero value at run-time, the NUSE and CCNT fields are set equal to zero. In addition, the TIMS and NSCS pointers are set to NULL. RES is then set back to zero.

The LVAL field contains the last value archived. The LTIM field contains the time in seconds at which the last value was archived. The CCNT keeps track of the array index. When a sample is archived, it's values and time stamps are written into the CCNT index of the appropriate arrays. The NUSE field keeps track of the number of samples that have been archived, while the NUSB field keeps track of the number of samples that have been written into the current buffers/

arrays. For example, if data for five samples has been written into the ARCH, VALS, TIMS, and NSCS fields, then NUSB will be five. The FTOF field is used as an offset when comparing the LTIM field with the current time. It will usually have the value of one.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ARCH	Archive Array	NOACCESS	No	NA	No	No	No	No
VALS	Values Array	NOACCESS	No	NA	No	No	Yes	No
TIMS	Times Array Seconds	NOACCESS	No	NA	No	No	Yes	No
NSCS	Times Array Nanoseconds	NOACCESS	No	NA	No	No	Yes	No
VAL	Array Value	NOACCESS	No	NA	Yes	No	No	No
TIM	Array Times Seconds	NOACCESS	No	NA	Yes	No	No	No
NSC	Array Times Nanoseconds	NOACCESS	No	NA	No	No	No	No
FLSH	Flush Flag	ULONG	No	0	Yes	No	No	No
RES	Reset	SHORT	No	0	Yes	Yes	No	No
LVAL	Last Value Stored	DOUBLE	No	0	Yes	No	No	No
LTIM	Last Time Monitored	ULONG	No	0	Yes	No	No	No
CCNT	Current Buffer Counter	ULONG	No	0	Yes	No	No	No
NUSE	Number Used	ULONG	No	0	Yes	No	No	No
NUSB	Number Used in Buffer	ULONG	No	0	Yes	No	No	No
FTOF	Flush Time Offset	ULONG	No	0	Yes	No	No	No

Chapter 7: bi - Binary Input

1. Introduction

The normal use for this record type is to obtain a binary value of 0 or 1. Most device support modules obtain values from hardware and place the value in RVAL. For these devices record processing sets VAL = (0,1) if RVAL is (0, not 0). Devices support modules may optionally read a value directly into VAL.

Soft device modules are provided to obtain input via database or channel access links or via dbPutField or dbPutLink requests. Two soft device support modules are provided—`Soft Channel` and `Raw Soft Channel`. The first allows VAL to be an arbitrary unsigned short integer. The second reads the value into RVAL just like normal hardware modules.

The binary input's fields fall into the following categories:

1. scan parameters
2. read and convert parameters
3. operator display parameters
4. alarm parameters
5. run-time parameters

2. Scan Parameters

The binary input record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Read and Convert Parameters

The read and convert fields determine from where the binary input gets its input from and how to convert the raw signal to engineering units. The INP field contains the address from where device support retrieves the value. If the binary input record gets its value from hardware, the address of the card must be entered in he INP field, and the name of the device support module must be entered in the DTYP field. See *Address Specification, Chapter 1, 2*, for information on the format of hardware addresses. Be aware that the format differs between types of cards. You can see a list of the device support modules currently supported at the user's local site by using the `dbst` utility (R3.13).

For records that specify the `Soft Channel` or `Raw Soft Channel` device support routines, the INP field can be a channel or database access link, or a constant. If a constant, VAL can be changed directly by dbPuts. See *Address Specification, Chapter 1, 2*, for information on the format of database and channel access addresses. Also, see Section 9.3, *Device Support for Soft Records*, in this chapter for more information on soft device support.

If the record gets its value from hardware or uses the `Raw Soft Channel` device support, the device support routines place the value into the RVAL field which is then converted using the process described in the next section.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
INP	Input Link	INLINK	Yes	0	No	No	N/A	No
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	
ZNAM	Zero Name	STRING [20]	Yes	Null	Yes	Yes	No	Yes
ONAM	One Name	STRING [20]	Yes	Null	Yes	Yes	No	Yes
RVAL	Raw Value	ULONG	No	0	Yes	Yes	Yes	Yes
VAL	Value Field	ENUM	No	0	Yes	Yes	Yes	Yes

3.1. Conversion Fields

The VAL field is set equal to (0, 1) if the RVAL field is (0, not 0), unless the device support module reads a value directly into VAL or the `Soft Channel` device support is used. The value can also be fetched as one of the strings specified in the ZNAM or ONAM fields. The ZNAM field has the string that corresponds to the 0 state, so when the value is fetched as this string, `put_enum_str` will return a 0 will. The ONAM field holds the string that corresponds to the 1 state, so when the value is fetched as this string, `put_enum_str` returns a 1.

ZNAM	ASCII string defining state zero
ONAM	ASCII string defining state one

4. Operator Display Parameters

These parameters are used to present meaningful data to the operator. The `get_enum_str` record support routine can retrieve the state string corresponding to VAL's state. So if the value is 1, `get_enum_str` will return the string in the ONAM field; and if 0, `get_enum_str` will return the ZNAM string.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ZNAM	Zero Name	STRING [20]	Yes	Null	Yes	Yes	No	Yes
ONAM	One Name	STRING [20]	Yes	Null	Yes	Yes	No	Yes
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

5. Alarm Parameters

These parameters are used to determine if the binary input is in alarm condition and to determine the severity of that condition. The possible alarm conditions for binary inputs are the SCAN, READ state alarms, and change of state alarm. The SCAN and READ alarms are called by the device support routines.

The user can choose the severity for each state in the ZSV and OSV. The possible values for these fields are NO ALARM, MINOR, and MAJOR. The ZSV holds the severity for the zero state; OSV, for the one state. COSV causes an alarm whenever the state changes between 0 and 1 and the severity is configured as MINOR or MAJOR.

See *Alarm Specification, Chapter 1, 4*, for a complete explanation of discrete alarm states. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ZSV	Zero Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
OSV	One Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
COSV	Change of State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes

6. Run-time Parameters and Simulation Mode Parameters

These parameters are used by the run-time code for processing the binary input. They are not configured using a database configuration tool.

ORAW is used to determine if monitors should be triggered for RVAL at the same time they are triggered for VAL.

MASK is given a value by the device support routines. This value is used to manipulate the record’s value, but is only the concern of the hardware device support routines.

The LALM field holds the value of the last occurrence of the change of state alarm. It is used to implement the change of state alarm, and thus only has meaning if COSV is MINOR or MAJOR.

The MLST is used by the `process` record support routine to determine if archive and value change monitors are invoked. They are if MLST is not equal to VAL.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ORAW	Old Raw Value	ULONG	No	0	Yes	No	No	No
MASK	Hardware mask	ULONG	No	compute	Yes	No	No	No
LALM	Last Alarmed value	USHORT	No	0	Yes	No	No	No
MLST	Last Monitored Value	USHORT	No	0	Yes	No	No	No

The following fields are used to operate the binary input in the simulation mode. See Chapter 3, *Fields Common to Many Record Types*, for more information on these fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No
SVAL	Simulation Value	DOUBLE	No	0	Yes	Yes	No	No
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

7. Record Support Routines

init_record

This routine initializes SIMM with the value of SIML if SIML type is CONSTANT link or creates a channel access link if SIML type is PV_LINK. SVAL is likewise initialized if SIOL is CONSTANT or PV_LINK.

This routine next checks to see that device support is available and a device support read routine is defined. If either does not exist, an error message is issued and processing is terminated.

If device support includes init_record, it is called.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_enum_str

Retrieves ASCII string corresponding to VAL.

get_enum_strs

Retrieves ASCII strings for ZNAM and ONAM.

put_enum_str

Checks if string matches ZNAM or ONAM, and if it does, sets VAL.

8. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. readValue is called. See Chapter 2, *Input Records*, for details.
3. If PACT has been changed to TRUE, the device support read routine has started but has not completed reading a new input value. In this case, the processing routine merely returns, leaving PACT TRUE.
4. Convert

```
    status=read_bi
    PACT = TRUE
    TIME = tslocaltime
    if status is 0, then set VAL=(0,1) if RVAL is (0, not
0)       and UDF = False
    if status is 2, set status = 0
```

5. Check alarms: This routine checks to see if the new VAL causes the alarm status and severity to change. If so, NSEV and NSTA and LALM are set. Note that if VAL is greater than 1, no checking is performed.
6. Check to see if monitors should be invoked:
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - Archive and value change monitors are invoked if MLST is not equal to VAL.
    - Monitors for RVAL are checked whenever other monitors are invoked.
    - NSEV and NSTA are reset to 0.
7. Scan forward link if necessary, set PACT FALSE, and return.

9. Device Support

9.1. Fields Of Interest To Device Support

Each input record must have an associated set of device support routines.

The primary responsibility of the device support routines is to obtain a new raw input value whenever read_bi is called. The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See Chapter 2, *Fields Common to All Record Types* for an explanation of these fields.
DPVT	Device Private	
UDF	VAL Undefined	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
VAL	Value Field	This field is set by a device support routines only if it doesn't want record support to set it.
INP	Input Link	This field is used by the device support routines to locate its input.
RVAL	Raw Value	It is the responsibility of the device support routine to give this field a value.
MASK	Hardware mask.	The device support routine must give this field a value if it needs to use it.

9.2. Device Support routines

Device support consists of the following routines:

report

```
report(FILE fp, paddr)
```

Not currently used.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
    *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

read_bi

```
read_bi(precord)
```

This routine must provide a new input value. It returns the following values:

- 0: Success. A new raw value is placed in RVAL. The record support module forces VAL to be (0,1) if RVAL is (0, not 0).
- 2: Success, but don't modify VAL.
- other: Error.

9.3. Device Support for Soft Records

Two soft device support modules Soft Channel and Raw Soft Channel are provided for input records not related to actual hardware devices. The INP link type must be either CONSTANT, DB_LINK, or CA_LINK.

Soft Channel

read_bi always returns a value of 2, which means that no conversion is performed.

If the INP link type is constant, then the constant value is stored into VAL by init_record, and UDF is set to FALSE. VAL can be changed via dbPut requests. If the INP link type is PV_LINK, then dbCaAddInlink is called by init_record.

read_bi calls recGblGetLinkValue to read the current value of VAL. See *Soft Input, Chapter 3, 2.3*, for details.

If the return status of recGblGetLinkValue is zero, then read_bi sets UDF to FALSE. The status of recGblGetLinkValue is returned.

Raw Soft Channel

This module is like the previous except that values are read into RVAL.

read_bi returns a value of 0. Thus the record processing routine will force VAL to be 0 or 1.

Chapter 8: bo—Binary Output

1. Introduction

The normal use for this record type is to store a simple bit (0 or 1) value to be sent to a Digital Output module. It can also be used to write binary values into other records via database or channel access links. This record can implement both latched and momentary binary outputs depending on how the HIGH field is configured.

The binary output's fields fall into the following categories:

scan parameters

convert and write parameters

operator display parameters

alarm parameters

run-time parameters

2. Scan Parameters

The binary output record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Desired Output Parameters

The binary output record must specify where its desired output originates. The desired output needs to be in engineering units.

The first field that determines where the desired output originates is the output mode select (OSML) field, which can have two possible values—`closed loop` or `supervisory`. If `supervisory` is specified, the value in the VAL field can be set externally via dbPuts at run-time. If `closed loop` is specified, the VAL field's value is obtained from the address specified in the desired output location (DOL) field which can be either a database link or a channel access link. To achieve continuous control, a database link to a control algorithm record should be entered in the DOL field. *Address Specification, Chapter 1, 2*, presents more information on database addresses and links. *Scanning Specification, Chapter 1, 1*, explains the effect of database linkage on scanning.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
DOL	Desired Output Location (Input Link)	INLINK	Yes	0	No	No	N/A	No
OMSL	Output Mode Select	GBLCHOICE	Yes	0	Yes	Yes	No	No

4. Convert and Write Parameters

These parameters are used to determine where the binary output writes to and how to convert the engineering units to a raw signal. After VAL is set and forced to be either 1 or 0, as the result of either a dbPut or a new value being retrieved from the link in the DOL field, then what happens next depends on which device support routine is used and how the HIGH field is configured.

If the `Soft Channel` device support routine is specified, then the device support routine writes the VAL field's value to the address specified in the OUT field. Otherwise, RVAL is the value finally written by the device support routines after being converted.

If VAL is equal to 0, then the record processing routine sets RVAL equal to zero. When VAL is not equal to 0, then RVAL is set equal to the value contained in the MASK field. (MASK is set by the device support routines and is of no concern to the user.) Also when VAL is not 0 and after RVAL is set equal to MASK, the record processing routine checks to see if the HIGH field is greater than 0. If it is, then the routine will process the record again with VAL set to 0 after the number of seconds specified by HIGH. Thus, HIGH implements a momentary output which changes the state of the device back to 0 after *N* number of seconds.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	
OUT	Output Link	OUTLINK	Yes	0	No	No	N/A	No
VAL	Value Field	ENUM	No	0	Yes	Yes	Yes	Yes
RVAL	Raw Data Value	ULONG	No	0	Yes	Yes	Yes	Yes
HIGH	Seconds to Hold High	FLOAT	Yes	0	Yes	Yes	No	No
ZNAM	Zero Name	STRING [20]	Yes	Null	Yes	Yes	No	Yes
ONAM	One Name	STRING [20]	Yes	Null	Yes	Yes	No	Yes

4.1. Conversion Parameters

The ZNAM field has the string that corresponds to the 0 state, and the ONAM field holds the string that corresponds to the 1 state. These fields, other than being used to tell the operator what each state represents, are used to perform conversions if the value fetched by DOL is a string. If it is, VAL is set to the state correspond to that string. For instance, if the value fetched is the string “Off” and the ZNAM string is “Off,” then VAL is set to 0.

After VAL is set, if VAL is equal to 0, then the record processing routine sets RVAL equal to zero. When VAL is not equal to 0, then RVAL is set equal to the value contained in the MASK field. (MASK is set by the device support routines and is of no concern to the user.) Also when VAL is equal to 1 and after RVAL is set equal to MASK, the record processing routine checks to see if the HIGH field is greater than 0. If it is, then the routine process the record again with VAL=0 after the number of seconds specified by HIGH. Thus, HIGH implements a latched output which changes the state of the device or link to 1, then changes it back to 0 after N number of seconds.

ZNAM	ASCII string defining state zero
ONAM	ASCII string defining state one
HIGH	If this value is greater than zero, then whenever VAL is set equal to 1, it is reset to zero after HIGH seconds.

4.2. Output Specification

The OUT field specifies where the binary output record gets its input. It must specify the address of an I/O card if the record sends its output to hardware, and the DTYP field must contain a the corresponding device support module. Be aware that the address format differs according to the I/O bus used. See *Address Specification, Chapter 1, 2*, for information on the format of hardware addresses. You can see a list of the device support modules currently supported at the user’s local site by using the `dbst` utility in R3.13.

Otherwise, if the record is configured to use the soft device support modules, then it can be either a database link, a channel access link, or a constant. Be aware that nothing will be written when OUT is a constant. See *Address Specification, Chapter 1, 2*, for information on the format of database and channel access addresses. Also, see Section 10.3, *Device Support For Soft Records*, in this chapter for more on output to other records.

5. Operator Display Parameters

These parameters are used to present meaningful data to the operator. The `get_enum_str` record support routine can retrieve the state string corresponding to the VAL’s state. So if the value is 1, `get_enum_str` will return the string in the ONAM field; and if 0, `get_enum_str` will return the ZNAM string.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ZNAM	Zero Name	STRING [20]	Yes	Null	Yes	Yes	No	Yes
ONAM	One Name	STRING [20]	Yes	Null	Yes	Yes	No	Yes

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

These parameters are used to determine the binary output's alarm condition and to determine the severity of that condition. The possible alarm conditions for binary outputs are the SCAN, READ , INVALID, and state alarms. The user can configure the state alarm conditions using these fields.

The possible values for these fields are `NO_ALARM`, `MINOR`, and `MAJOR`. The ZSV holds the severity for the zero state; OSV for the one state. COSV is used to cause an alarm whenever the state changes between the states (0-1, or 1-0) and its severity is configured as MINOR or MAJOR.

See *Invalid Alarm Output Action, Chapter 3, 3.5*, for more information on the IVOA and IVOV fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ZSV	Zero Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
OSV	One Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
COSV	Change of State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
IVOA	Invalid Alarm	GBLCHOICE	Yes	0	Yes	Yes	No	No
IVOV	Invalid Alarm Output Value	USHORT	Yes	0	Yes	Yes	No	No

7. Run-Time and Simulation Mode Parameters

These parameters are used by the run-time code for processing the binary output. They are not configurable using a configuration tool. They represent the current state of the binary output.

ORAW is used to determine if monitors should be triggered for RVAL at the same time they are triggered for VAL.

MASK is given a value by the device support routines and should not concern the user.

The RBV field is also set by device support. It is the actual read back value obtained from the hardware itself or from the associated device driver. The ORBV field is used to decide if monitors should be triggered for RBV at the same time monitors are triggered for changes in VAL.

The LALM field holds the value of the last occurrence of the change of state alarm. It is used to implement the change of state alarm, and thus only has meaning if COSV is MINOR or MAJOR.

The MLST is used by the `process` record support routine to determine if archive and value change monitors are invoked. They are if MLST is not equal to VAL.

The WPDT field is a private field for honoring seconds to hold HIGH.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ORAW	Old Raw Value	ULONG	No	0	Yes	No	No	No
MASK	Hardware Mask	ULONG	No	compute	Yes	No	No	No
RBV	Readback Value	ULONG	No	0	Yes	No	No	No
ORBV	Old Readback Value	ULONG	No	0	Yes	No	No	No
LALM	Last Alarmed Value	USHORT	No	0	Yes	No	No	No
MLST	Last Monitored Value	USHORT	No	0	Yes	No	No	No
RPVT	Record Private	NOACCESS	No	0	No	No	No	No
WDPT	Watchdog Pointer	NOACCESS	No	0	No	No	No	No

The following fields are used to operate the binary output in the simulation mode. See Chapter 3, *Fields Common to Many Record Types*, for more information on these fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No
SVAL	Simulation Value	DOUBLE	No	0	Yes	Yes	No	No
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

8. Record Support Routines

init_record

This routine initializes SIMM if SIML is a constant or creates a channel access link if SIML is PV_LINK. If SIOL is PV_LINK a channel access link is created .

This routine next checks to see that device support is available. The routine next checks to see if the device support write routine is defined.

If either device support or the device support write routine does not exist, an error message is issued and processing is terminated.

If DOL is a constant, then VAL is initialized to 1 if its value is nonzero or initialized to 0 if DOL is zero, and UDF is set to FALSE.

If device support includes init_record, it is called. VAL is set using RVAL, and UDF is set to FALSE.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_enum_str

Retrieves ASCII string corresponding to VAL.

get_enum_strs

Retrieves ASCII strings for ZNAM and ONAM.

put_enum_str

Checks if string matches ZNAM or ONAM, and if it does, sets VAL.

9. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. If PACT is FALSE
   - if DOL is DB_LINK and OMSL is CLOSED_LOOP
     - get value from DOL
     - check for link alarm
     - force VAL to be 0 or 1
     - if MASK is defined
       - if VAL is 0 set RVAL = 0
     - else set RVAL = MASK
3. Check alarms: This routine checks to see if the new VAL causes the alarm status and severity to change. If so, NSEV, NSTA and LALM are set.
4. Check severity and write the new value. See *Invalid Alarm Output Action, Chapter 3, 3.5*, for more information on how INVALID alarms affect output.

5. If PACT has been changed to TRUE, the device support write output routine has started but has not completed writing the new value. In this case, the processing routine merely returns, leaving PACT TRUE.
6. Check WAIT. If VAL is 1 and WAIT is greater than 0, process again with a VAL=0 after WAIT seconds.
7. Check to see if monitors should be invoked.
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - Archive and value change monitors are invoked if MLST is not equal to VAL.
    - Monitors for RVAL and for RBV are checked whenever other monitors are invoked.
    - NSEV and NSTA are reset to 0.
8. Scan forward link if necessary, set PACT FALSE, and return.

10. Device Support

10.1. Fields Of Interest To Device Support

Each binary output record must have an associated set of device support routines. The primary responsibility of the device support routines is to write a new value whenever write_bo is called. The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See Chapter 2, *Fields Common to All Record Types* for an explanation of these fields.
DPVT	Device Private	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
VAL	Value Field	This field is only of interest to device support routines that do not use MASK and RVAL.
OUT	Output Link	This field is used by the device support routines to locate its output.
RVAL	Raw Data Value	If MASK is defined then record support sets RVAL=(0,MASK) if VAL is (0, not zero).
MASK	Hardware mask.	The device support module must set this field. Not that if VAL is 1, then record processing sets RVAL = MASK.
RBV	Read Back Value	This is the actual read back value obtained from the hardware itself or from the associated device driver. It is the responsibility of the device support routine to give this field a value.

10.2. Device Support routines

Device support consists of the following routines:

report

```
report(FILE fp, paddr)
```

Not currently used.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine. It should determine MASK if it is needed.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
    *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

write_bo

```
write_bo(precord)
```

This routine must output an new value. It returns the following values:

- 0: Success.
- other: Error.

10.3. Device Support For Soft Records

Two soft device support modules `Soft Channel` and `Raw Soft Channel` are provided for output records not related to actual hardware devices. The OUT link type must be either a CONSTANT, DB_LINK, or CA_LINK.

Soft Channel

This module writes the current value of VAL.

If the OUT link type is PV_LINK, then dbCaAddInlink is called by init_record. init_record always returns a value of 2, which means that no conversion will ever be attempted. write_bo calls recGblPutLinkValue to write the current value of VAL. See *Soft Output, Chapter 3, 3.2*, for details.

Raw Soft Channel

This module is like the previous except that it writes the current value of RVAL.

Chapter 9: Calc - Calculation

1. Introduction

The calculation or “Calc” record is used to perform algebraic, relational, and logical operations on values retrieved from other records. The result of its operations can then be accessed by another record so that it can be used. The fields in this record fall into these categories:

scan parameters

read parameters

expression parameters

operator display parameters

alarm parameters

monitor parameters

run-time parameters

2. Scan Parameters

The Calc record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Since the Calc record supports no direct interfaces to hardware, it cannot be scanned on I/O interrupt, so its SCAN field cannot be `I/O Intr`.

3. Read Parameters

The read parameters for the Calc record consist of 12 input links—INPA, INPB, . . . INPL. The fields can be database links, channel access links, or constants. If they are links, they must specify another record’s field or a channel access link. If they are constants, they will be initialized with the value they are configured with and can be changed via `dbPuts`. They cannot be hardware addresses.

See *Address Specification, Chapter 1, 2*, for information on how to specify database links.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor
INPA	Input Link A	INLINK	Yes	0	No	No	N/A
INPB	Input Link B	INLINK	Yes	0	No	No	N/A
INPC	Input Link C	INLINK	Yes	0	No	No	N/A
INPD	Input Link D	INLINK	Yes	0	No	No	N/A
INPE	Input Link E	INLINK	Yes	0	No	No	N/A
INPF	Input Link F	INLINK	Yes	0	No	No	N/A
INPG	Input Link G	INLINK	Yes	0	No	No	N/A
INPH	Input Link H	INLINK	Yes	0	No	No	N/A
INPI	Input Link I	INLINK	Yes	0	No	No	N/A
INPJ	Input Link J	INLINK	Yes	0	No	No	N/A
INPK	Input Link K	INLINK	Yes	0	No	No	N/A
INPL	Input Link L	INLINK	Yes	0	No	No	N/A

4. Expression

At the core of the Calc record lie the CALC and RPCL fields. The CALC field contains the infix expression which the record routine will use when it processes the record. The resulting value is placed in the VAL field and can be accessed from there. The CALC expression is actually converted to opcode and stored as in Reverse Polish Notation in the RPCL field. It is this expression which is actually used to calculate VAL. The Reverse Polish expression is evaluated more efficiently during run-time than an infix expression. CALC can be changed at run-time, and a special record routine calls a function to convert it to Reverse Polish Notation.

The range of expressions supported by the calculation record are separated into operands, algebraic operators, trigonometric operators, relational operators, logical operators, parentheses and commas, and the question mark or ‘?:’ operator. The expression can consist of any of these operators, as well as any of the values from the input links which are the operands.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
CALC	Calculation	DBF_STRING	Yes	0	Yes	Yes	Yes	Yes
RPCL	Reverse Polish	DBF_NOACCESS	No	0	No	No	N/A	No

4.1. Operands

The expression uses the values retrieved from the INP*x* links as operands, though constants can be used as operands too. These values retrieved from the input links are stored in the A-L fields. The values to be used in the expression are simply referenced by the field letter. For instance, the value obtained from the INPA link is stored in the field A, and the value obtained from INPB is stored in field B. The field names can be included in the expression which will operate on their respective values, as in A+B. Also, the RNDM unary function can be included as an operand in the expression in order to generate a random number between 0 and 1.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
A	Input Value A	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
B	Input Value B	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
C	Input Value C	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
D	Input Value D	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
E	Input Value E	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
F	Input Value F	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
G	Input Value G	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
H	Input Value H	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
I	Input Value I	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
J	Input Value J	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
K	Input Value K	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
L	Input Value L	DOUBLE	No	0	Yes	Yes/No	Yes	Yes

4.2. Algebraic Operators

- ABS: Absolute value (unary)
- SQR: Square root (unary)
- MIN: Minimum (binary function)
- MAX: Maximum (binary function)
- CEIL: Ceiling (unary)
- FLOOR: Floor (unary)
- LOG: Log base 10 (unary)
- LOGE: Natural log (unary)
- EXP: Exponential function (unary)
- ^: Exponential (binary)
- ** : Exponential (binary)
- + : Addition (binary)
- - : Subtraction (binary)
- * : Multiplication (binary)

- / : Division (binary)
- % : Modulo (binary)
- NOT: Negate (unary)

4.3. Trigonometric Operators

- SIN: Sine
- SINH: Hyperbolic sine
- ASIN: Arc sine
- COS: Cosine
- COSH: Hyperbolic cosine
- ACOS: Arc cosine
- TAN: Tangent
- TANH: Hyperbolic tangent
- ATAN: Arc tangent

4.4. Relational Operators

- >= : Greater than or equal to
- > : Greater than
- <= : Less than or equal to
- <: Less than
- # : Not equal to
- = : Equal to

4.5. Logical Operators

- && : And
- || : Or
- ! : Not

4.6. Bitwise Operators

- | : Bitwise Or
- & : Bitwise And
- OR : Bitwise Or
- AND: Bitwise And
- XOR: Bitwise Exclusive Or
- ~ : One's Complement
- << : Left shift
- >> : Right shift

4.7. Parentheses and Comma

The open and close parentheses are supported. Nested parenthesis are supported.

The comma is supported when used to separate the arguments of a binary function.

4.8. Conditional Expression

The C language's question mark operator is supported. The format is:

```
(condition)? True result : False result
```

4.9. Examples

Algebraic

```
A + B + 10
```

•Result is A + B

Relational

```
(A + B) < (C + D)
```

•Result is 1 if (A+B) < (C+D)

•Result is 0 if (A+B) >= (C+D)

Question Mark

```
(A+B)<(C+D)?E:F+L+10
```

•Result is E if (A+B) < (C+D)

•Result is F+L+10 if (A+B) >= (C+D)

```
(A+B)<(C+D)?E
```

•Result is E if (A+B) < (C+D)

•Result is unchanged if (A+B) >= (C+D)

Logical

```
A&B
```

•Causes the following to occur:

- Convert A to integer
- Convert B to integer
- Bit-wise And A and B
- Convert result to floating point

5. Operator Display Parameters

These parameters are used to present meaningful data to the operator. These fields are used to display VAL and the other parameters of the calculation record either textually or graphically.

The EGU field contains a string of up to 16 characters which is supplied by the user and which describes the values being operated upon. The string is retrieved whenever the routine `get_units` is called. The EGU string is solely for an operator's sake and does not have to be used.

The HOPR and LOPR fields only refer to the limits of the VAL, HIHI, HIGH, LOW, and LOLO fields. PREC controls the precision of the VAL field.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	Null	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

The possible alarm conditions for the Calc record are the SCAN, READ, Calculation, and limit alarms. The SCAN and READ alarms are called by the record support routines. The Calculation alarm is called by the record processing routine when the CALC expression is an invalid one, upon which an error message is generated.

The following alarm parameters which are configured by the user define the limit alarms for the VAL field and the severity corresponding to those conditions.

The HYST field defines an alarm deadband for each limit. See *Alarm Specification, Chapter 1, 4*, for a complete explanation of alarms and these fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HIHI	Hihi Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HIGH	High Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOW	Low Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOLO	Lolo Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HHSV	Severity for a Hihi Alarm	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HSV	Severity for a High Alarm	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LSV	Severity for a Low Alarm	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LLSV	Severity for a Lolo Alarm	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HYST	Alarm Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

7. Monitor Parameters

These parameters are used to determine when to send monitors for the value fields. The monitors are sent when the value field exceeds the last monitored field by the appropriate deadband, the ADEL for archiver monitors and the MDEL field for all other types of monitors. If these fields have a value of zero, everytime the value changes, monitors are triggered; if they have a value of -1, everytime the record is scanned, monitors are triggered. See *Monitor Specification, Chapter 1, 5*, for a complete explanation of monitors.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ADEL	Archive Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
MDEL	Monitor, i.e. value change, Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

8. Run-time Parameters

These fields are not configurable using a configuration too and none are modifiable at run-time. They are used to process the record.

The LALM field is used to implement the hysteresis factor for the alarm limits.

The LA-LL fields are used to decide when to trigger monitors for the corresponding fields. For instance, if LA does not equal the value for A, monitors for A are triggered. The MLST and MLST fields are used in the same manner for the VAL field.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LALM	Last Alarmed Value	DOUBLE	No	0	Yes	No	No	No
ALST	Archive Last Value	DOUBLE	No	0	Yes	No	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
MLST	Monitor Last Value	DOUBLE	No	0	Yes	No	No	No
LA	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LB	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LC	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LD	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LE	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LF	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LG	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LH	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LI	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LJ	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LK	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LL	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No

9. Record Support Routines

init_record

For each constant input link, the corresponding value field is initialized with the constant value if the input link is CONSTANT or a channel access link is created if the input link is PV_LINK.

A routine postfix is called to convert the infix expression in CALC to reverse polish notation. The result is stored in RPCL.

process

See next section.

special

This is called if CALC is changed. special calls postfix.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_units

Retrieves EGU.

get_precision

Retrieves PREC.

get_graphic_double

Sets the upper display and lower display limits for a field. If the field is VAL, HIHI, HIGH, LOW, or LOLO, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_control_double

Sets the upper control and the lower control limits for a field. If the field is VAL, HIHI, HIGH, LOW, or LOLO, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_alarm_double

Sets the following values:

upper_alarm_limit = HIHI

upper_warning_limit = HIGH

lower_warning_limit = LOW

lower_alarm_limit = LOLO

10. Record Processing

Routine process implements the following algorithm:

1. Fetch all arguments.
2. Call routine calcPerform, which calculates VAL from the postfix version of the expression given in CALC. If calcPerform returns success UDF is set to FALSE.
3. Check alarms. This routine checks to see if the new VAL causes the alarm status and severity to change. If so, NSEV, NSTA and LALM are set. It also honors the alarm hysteresis factor (HYST). Thus the value must change by at least HYST before the alarm status and severity changes.
4. Check to see if monitors should be invoked.
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - Archive and value change monitors are invoked if ADEL and MDEL conditions are met.
    - Monitors for A-L are checked whenever other monitors are invoked.
    - NSEV and NSTA are reset to 0.

5. Scan forward link if necessary, set PACT FALSE, and return.

Chapter 10:Calcout - Calculation Output Record

1. Introduction

The Calculation Output or “Calcout” record is similar to the Calc record with the added feature of having outputs (an "output link" and an "output event") which are conditionally executed based on the result of the calculation. This feature allows conditional branching to be implemented within an EPICS database (e.g., process Record_A only if Record_B has a value of 0). The Calcout record is also similar to the Wait record (with additional features) but uses EPICS standard INLINK and OUTLINK fields rather than the DBF_STRING fields used in the Wait record. For new databases, it is recommended that the Calcout record be used instead of the Wait record. The fields in this record fall into these categories:

scan parameters

read parameters

expression parameters

output parameters

operator display parameters

alarm parameters

monitor parameters

run-time parameters

2. Scan Parameters

The Calcout record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Since the Calcout record supports no direct interfaces to hardware, it cannot be scanned on I/O interrupt, so its SCAN field cannot be `I/O Intr`.

3. Read Parameters

The read parameters for the Calcout record consist of 12 input links—INPA, INPB, . . . INPL. The fields can be database links, channel access links, or constants. If they are links, they must specify another record's field. If they are constants, they will be initialized with the value they are configured with and can be changed via `dbPuts`. These fields cannot be hardware addresses. In addition, the Calcout record contains the INAV, INBV, . . . INLV fields which indicate the status of the link fields, for example, whether or not the specified PV was found and a link to it established. See Section 6, *Operator Display Parameters* for an explanation of these fields.

See *Address Specification, Chapter 1, 2*, for information on how to specify database links.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor
INPA	Input Link A	INLINK	Yes	0	No	No	N/A
INPB	Input Link B	INLINK	Yes	0	No	No	N/A
INPC	Input Link C	INLINK	Yes	0	No	No	N/A
INPD	Input Link D	INLINK	Yes	0	No	No	N/A
INPE	Input Link E	INLINK	Yes	0	No	No	N/A
INPF	Input Link F	INLINK	Yes	0	No	No	N/A
INPG	Input Link G	INLINK	Yes	0	No	No	N/A
INPH	Input Link H	INLINK	Yes	0	No	No	N/A
INPI	Input Link I	INLINK	Yes	0	No	No	N/A
INPJ	Input Link J	INLINK	Yes	0	No	No	N/A
INPK	Input Link K	INLINK	Yes	0	No	No	N/A
INPL	Input Link L	INLINK	Yes	0	No	No	N/A

4. Expression

Like the Calc record, the Calcout record has a CALC field in which the developer can enter an infix expression which the record routine will evaluate when it processes the record. The resulting value is placed in the VAL field. This value can then be used by the OOPT field (see Section 5, *Output Parameters*) to determine whether or not to write to the output link or post an output event. It can also be the value that is written to the output link. The CALC expression is actually converted to opcode and stored in Reverse Polish Notation in the RPCL field. It is this expression which is actually used to calculate VAL. The Reverse Polish expression is evaluated more efficiently during run-time than an infix expression. CALC can be changed at run-time, and a special record routine will call a function to convert it to Reverse Polish Notation.

The range of expressions supported by the calculation record are separated into operands, algebraic operators, trigonometric operators, relational operators, logical operators, parentheses and commas, and the conditional '?:' operator. The expression can consist of any of these operators, as well as any of the values from the input links which are the operands. Also, the RNDM unary function can be included as an operand in the expression in order to generate a random number between 0 and 1.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
CALC	Calculation	STRING[36]	Yes	0	Yes	Yes	Yes	No
VAL	Value	DOUBLE	No	0	Yes	Yes	Yes	No
RPCL	Reverse Polish	NOACCESS	No	0	No	No	N/A	No

4.1. Operands

The expression can use the values retrieved from the INP*x* links as operands, though constants can be used as operands too. These values retrieved from the input links are stored in the A-L fields. The values to be used in the expression are simply referenced by the field letter. For instance, the value obtained from the INPA link is stored in the field A, and the value obtained from INPB is stored in field B. The field names can be included in the expression which will operate on their respective values, as in A+B.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
A	Input Value A	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
B	Input Value B	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
C	Input Value C	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
D	Input Value D	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
E	Input Value E	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
F	Input Value F	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
G	Input Value G	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
H	Input Value H	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
I	Input Value I	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
J	Input Value J	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
K	Input Value K	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
L	Input Value L	DOUBLE	No	0	Yes	Yes/No	Yes	Yes

4.2. Algebraic Operators

- ABS: Absolute value (unary)
- SQR: Square root (unary)
- MIN: Minimum (binary function)
- MAX: Maximum (binary function)
- CEIL: Ceiling (unary)
- FLOOR: Floor (unary)
- LOG: Log base 10 (unary)
- LOGE: Natural log (unary)
- EXP: Exponential function (unary)
- ^: Exponential (binary)
- ** : Exponential (binary)
- + : Addition (binary)
- - : Subtraction (binary)
- * : Multiplication (binary)
- / : Division (binary)
- % : Modulo (binary)
- NOT: Negate (unary)

4.3. Trigonometric Operators

- SIN: Sine
- SINH: Hyperbolic sine
- ASIN: Arc sine
- COS: Cosine
- COSH: Hyperbolic cosine
- ACOS: Arc cosine
- TAN: Tangent
- TANH: Hyperbolic tangent
- ATAN: Arc tangent

4.4. Relational Operators

- >= : Greater than or equal to
- > : Greater than
- <= : Less than or equal to
- <: Less than
- # : Not equal to
- = : Equal to

4.5. Logical Operators

- && : And
- || : Or
- ! : Not

4.6. Bitwise Operators

- | : Bitwise Or
- & : Bitwise And
- OR : Bitwise Or
- AND: Bitwise And
- XOR: Bitwise Exclusive Or
- ~ : One's Complement
- << : Left shift
- >> : Right shift

4.7. Parentheses and Comma

The open and close parentheses are supported. Nested parenthesis are supported.

The comma is supported when used to separate the arguments of a binary function.

4.8. Conditional Expression

The C language’s question mark operator is supported. The format is:

```
(condition)? True result : False result
```

4.9. Examples

Algebraic

```
A + B + 10
```

- Result is A + B

Relational

```
(A + B) < (C + D)
```

- Result is 1 if (A+B) < (C+D)
- Result is 0 if (A+B) >= (C+D)

Question Mark

`(A+B)<(C+D)?E:F+L+10`

•Result is E if (A+B) < (C+D)

•Result is F+L+10 if (A+B) >= (C+D)

`(A+B)<(C+D)?E`

•Result is E if (A+B) < (C+D)

•Result is unchanged if (A+B) >= (C+D)

Logical

`A&B`

•Causes the following to occur:

- Convert A to integer
- Convert B to integer
- Bit-wise And A and B
- Convert result to floating point

5. Output Parameters

These parameters specify and control the output capabilities of the Calcout record. They determine when to write the output, where to write it, and what the output will be. The OUT link specifies the Process Variable to which the result will be written. The OOPT field determines the condition that causes the output link to be written to. It's a menu field that has six choices:

•`Every Time` — write output every time record is processed.

•`On Change` — write output every time VAL changes, i.e., every time the result of the expression changes.

•`When Zero` — when record is processed, write output if VAL is zero.

•`When Non-zero` — when record is processed, write output if VAL is non-zero.

•`Transition to Zero` — when record is processed, write output only if VAL is zero and last value was non-zero.

•`Transition to Non-zero` — when record is processed, write output only if VAL is non-zero and last value was zero.

The DOPT field determines what data is written to the output link when the output is executed. The field is a menu field with two options: `Use CALC` or `Use OCAL`. If `Use CALC` is specified, when the record writes its output it will write the result of the expression in the CALC record, that is, it will write the value of the VAL field. If `Use OCAL` is specified, the record will instead write the result of the expression in the OCAL field, which is contained in the OVAL field. The OCAL field is exactly like the CALC field and has the same functionality: it can contain the string representation of an expression which is evaluated at run-time. Thus, if necessary, the record can use the result of the CALC expression to determine if data should be written and can use the result of the OCAL expression as the data to write.

If the OEVT field specifies a non-zero integer and the condition in the OOPT field is met, the record will post a corresponding event. If the ODLY field is non-zero, the record pauses for the specified number of seconds before executing the OUT link or posting the output event. During this waiting period the record is "active" and will not be processed again until the wait is over. The field DLYA is equal to 1 during the delay period. The resolution of the delay entry is 1/60 of a second (it uses the VxWorks `tickLib`).

The IVOA field specifies what action to take with the OUT link if the Calcout record enters an INVALID alarm status. The options are `Continue normally`, `Don't drive outputs`, and `Set output to IVOV`. If the IVOA field is `Set output to IVOV`, the data entered into the IVOV field is written to the OUT link if the record alarm severity is INVALID.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OUT	Output Specification	OUTLINK	Yes	0	Yes	Yes	N/A	No
OOPT	Output Execute Option	RECCHOICE	Yes	0	Yes	Yes	No	No
DOPT	Output Data Option	RECCHOICE	Yes	0	Yes	Yes	No	No
OCAL	Output Calculation	STRING[36]	Yes	Null	Yes	Yes	No	No
OVAL	Output Value	DOUBLE	No	0	Yes	Yes	Yes	Yes
OEVT	Event To Issue	SHORT	Yes	0	Yes	Yes	No	No
ODLY	Output Execution Delay	FLOAT	Yes	0	Yes	Yes	No	No
IVOV	Invalid Output Action	GBLCHOICE	Yes	0	Yes	Yes	No	No
IVOA	Invalid Output Value	DOUBLE	Yes	0	Yes	Yes	No	No

6. Operator Display Parameters

These parameters are used to present meaningful data to the operator. Some are also meant to represent the status of the record at run-time. An example of an interactive MEDM display screen that displays the status of the Calcout record is located here.

The EGU field contains a string of up to 16 characters which is supplied by the user and which describes the values being operated upon. The string is retrieved whenever the routine `get_units` is called. The EGU string is solely for an operator's sake and does not have to be used.

The HOPR and LOPR fields only refer to the limits of the VAL, HIHI, HIGH, LOW, and LOLO fields. PREC controls the precision of the VAL field.

The INAV-INLV fields indicate the status of the link to the PVs specified in the INPA-INPL fields, respectively. The field can have three possible values:

- `Ext PV NC` — the PV wasn't found on this IOC and a Channel Access link hasn't been established.
- `Ext PV OK` — the PV wasn't found on this IOC and a Channel Access link has been established.
- `Local PV` — the PV was found on this IOC.
- `Constant` — the corresponding link field is a constant.

The OUTV field indicates the status of the OUT link. It has the same possible values as the INAV-INLV fields.

The CLCV and OLCV fields indicate the validity of the expression in the CALC and OCAL fields, respectively. If the expression is invalid, the field is set to one.

The DLYA field is set to one during the delay interval specified in ODLY.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	Null	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
INAV	Link Status of INPA	RECCHOICE	No	1	Yes	No	No	No
INBV	Link Status of INPB	RECCHOICE	No	1	Yes	No	No	No
INCV	Link Status of INPC	RECCHOICE	No	1	Yes	No	No	No
INDV	Link Status of INPD	RECCHOICE	No	1	Yes	No	No	No
INEV	Link Status of INPE	RECCHOICE	No	1	Yes	No	No	No
INFV	Link Status of INPF	RECCHOICE	No	1	Yes	No	No	No
INGV	Link Status of INPG	RECCHOICE	No	1	Yes	No	No	No
INHV	Link Status of INPH	RECCHOICE	No	1	Yes	No	No	No
INIV	Link Status of INPI	RECCHOICE	No	1	Yes	No	No	No
INJV	Link Status of INPJ	RECCHOICE	No	1	Yes	No	No	No
INKV	Link Status of INPK	RECCHOICE	No	1	Yes	No	No	No
INLV	Link Status of INPL	RECCHOICE	No	1	Yes	No	No	No
OUTV	OUT PV Status	RECCHOICE	No	0	Yes	No	No	No
CLCV	CALC Valid	LONG	No	0	Yes	Yes	No	No
OCLV	OCAL Valid	LONG	No	0	Yes	Yes	No	No
DLYA	Output Delay Active	USHORT	No	0	Yes	No	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

7. Alarm Parameters

The possible alarm conditions for the Calc record are the SCAN, READ, Calculation, and limit alarms. The SCAN and READ alarms are called by the record support routines. The Calculation alarm is called by the record processing routine when the CALC expression is an invalid one, upon which an error message is generated.

The following alarm parameters which are configured by the user define the limit alarms for the VAL field and the severity corresponding to those conditions.

The HYST field defines an alarm deadband for each limit. See *Alarm Specification, Chapter 1, 4*, for a complete explanation of alarms and these fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HIHI	Hihi Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HIGH	High Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOW	Low Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOLO	Lolo Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HHSV	Severity for a Hihi Alarm	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HSV	Severity for a High Alarm	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LSV	Severity for a Low Alarm	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LLSV	Severity for a Lolo Alarm	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HYST	Alarm Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

8. Monitor Parameters

These parameters are used to determine when to send monitors for the value fields. The monitors are sent when the value field exceeds the last monitored field by the appropriate deadband, the ADEL for archiver monitors and the MDEL field for all other types of monitors. If these fields have a value of zero, every time the value changes, monitors are triggered; if they have a value of -1, every time the record is scanned, monitors are triggered. See *Monitor Specification, Chapter 1, 5*, for a complete explanation of monitors.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ADEL	Archive Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
MDEL	Monitor, i.e. value change, Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

9. Run-time Parameters

These fields are not configurable using a configuration too and none are modifiable at run-time. They are used to process the record.

The LALM field is used to implement the hysteresis factor for the alarm limits.

The LA-LL fields are used to decide when to trigger monitors for the corresponding fields. For instance, if LA does not equal the value for A, monitors for A are triggered. The MLST and MLST fields are used in the same manner for the VAL field.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LALM	Last Alarmed Value	DOUBLE	No	0	Yes	No	No	No
ALST	Archive Last Value	DOUBLE	No	0	Yes	No	No	No
MLST	Monitor Last Value	DOUBLE	No	0	Yes	No	No	No
LA	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LB	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LC	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LD	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LE	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LF	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LG	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LH	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LI	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LJ	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LK	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No
LL	Previous Input Value for A	DOUBLE	No	0	Yes	No	No	No

10. Record Support Routines

init_record

For each constant input link, the corresponding value field is initialized with the constant value if the input link is CONSTANT or a channel access link is created if the input link is PV_LINK.

A routine postfix is called to convert the infix expression in CALC and OCAL to reverse polish notation. The result is stored in RPCL and ORPC, respectively.

process

See next section.

special

This is called if CALC or OCAL is changed. special calls postfix.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_units

Retrieves EGU.

get_precision

Retrieves PREC.

get_graphic_double

Sets the upper display and lower display limits for a field. If the field is VAL, HIHI, HIGH, LOW, or LOLO, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_control_double

Sets the upper control and the lower control limits for a field. If the field is VAL, HIHI, HIGH, LOW, or LOLO, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_alarm_double

Sets the following values:

upper_alarm_limit = HIHI

upper_warning_limit = HIGH

lower_warning_limit = LOW

lower_alarm_limit = LOLO

11. Record Processing

11.1. process()

The `process()` routine implements the following algorithm:

1. Fetch all arguments.
2. Call routine calcPerform(), which calculates VAL from the postfix version of the expression given in CALC. If calcPerform() returns success, UDF is set to FALSE.
3. Check alarms. This routine checks to see if the new VAL causes the alarm status and severity to change. If so, NSEV, NSTA and LALM are set. It also honors the alarm hysteresis factor (HYST). Thus the value must change by at least HYST before the alarm status and severity changes.
4. Determine if the Output Execution Option (OOPT) is met. If it is met, either execute the output link (and output event) immediately (if ODLY = 0), or schedule a callback after the specified interval. See the explanation for the `execOutput()` routine below.
5. Check to see if monitors should be invoked.
   - Alarm monitors are invoked if the alarm status or severity has changed.
   - Archive and value change monitors are invoked if ADEL and MDEL conditions are met.
   - Monitors for A-L are checked whenever other monitors are invoked.
   - NSEV and NSTA are reset to 0.
6. If no output delay was specified, scan forward link if necessary, set PACT FALSE, and return.

11.2. execOutput()

1. If DOPT field specifies the use of OCAL, call the routine calcPerform for the postfix version of the expression in OCAL. Otherwise, use VAL.
2. If the Alarm Severity is INVALID, follow the option as designated by the field IVOA.
3. If the Alarm Severity is not INVALID or IVOA specifies "Continue Normally", put the value of OVAL to the OUT link and post the event in OEVT (if non-zero).
4. If an output delay was implemented, process the forward link.

Chapter 11:compress - Compression

1. Introduction

The data compression record is used to collect and compress data from arrays. When the INP field references a data array field, it immediately compresses the entire array into an element of an array using one of several algorithms, overwriting the previous element. If the INP field obtains its value from a scalar-value field, the compression record will collect a new sample each time the record is processed and add it to the compressed data array as a circular buffer.

The INP link can also specify a constant; however, if this is the case, the compression algorithms are ignored, and the record support routines merely return after checking the FLNK field.

The data compression fields fall into the following categories:

- scan parameters
- read parameters
- operator display parameters
- run-time parameters

2. Scanning Parameters

The compression record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Since the compression record supports no direct interfaces to hardware, its SCAN field cannot specify `I/O Intr`.

3. Read Parameters and Algorithm Parameters

These fields determine what channel to read and how to compress the data. The user specifies the algorithm to be used in the ALG field. There are five possible algorithms which can be specified as follows:

```
Circular Buffer
Average
N to 1 Low Value
```

`N to 1 High Value`

`N to 1 Average`

These algorithms are explained in one of the sections below.

The RES field can be accessed at run time to cause the algorithm to reset itself.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
RES	Reset	SHORT	No	0	Yes	Yes	No	No
ALG	Algorithm	RECCHOICE	Yes	0	Yes	No	No	No
NSAM	Number in Sample	ULONG	Yes	1	Yes	No	No	No
N	Number	ULONG	Yes	1	Yes	No	No	No
ILIL	Initial Low Interest Value	FLOAT	Yes	0	Yes	Yes	No	No
IHIL	Initial High Interest Value	FLOAT	Yes	0	Yes	Yes	No	No

3.1. Input Specification

The input specification should be a database or channel access link. Though INP can be a constant, the data compression algorithms are supported only when INP is a database link. See *Address Specification, Chapter 1, 2*, for information on specifying links.

3.2. Algorithms and Related Fields

As stated above, the ALG field specifies which algorithm to be performed on the data. The rest of the fields—NSAM, N, ILIL, IHIL, and OFF—are used in the compressions, though N is not used in the `Circular Buffer` algorithm and ILIL, IHIL, and OFF are used neither in the `Circular Buffer` algorithm nor in the `Average` algorithm.

The `Circular Buffer` algorithm keeps a circular buffer of length NSAM. Each time the record is processed, it gets the data referenced by INP and puts it into the circular buffer referenced by VAL. Note that when INP refers to a scalar, VAL is just a time ordered circular buffer of values obtained from INP.

`Average` takes an average of all the elements of the array obtained from INP; that is, the entire array referenced by INP is retrieved, and the average of the elements is placed in the next element of the circular buffer. The retrieved array is truncated to be of length NSAM. N successive arrays are averaged and placed in the circular buffer. Thus, VAL[0] holds the average of the first sampling of INP, VAL[1] holds the average of the next sampling of INP, and so on. The following sums up the equation:

$$\mathrm{VAL}\,[i] \leftarrow \frac{1}{N}\sum \mathrm{INP}\,[i]$$

If `N to 1 Low Value`, `N to 1 High Value`, or `N to 1 Average` are chosen, then VAL is a circular buffer of NSAM samples. The actual algorithm depends on whether INP references a scalar or an array. If INP refers to a scalar, then N successive time ordered samples of INP are taken. After the Nth sample is obtained a new value, determined by the algorithm (LOW, HIGH, or AVE), is written to the circular buffer referenced by VAL. If LOW, the lowest value is written; if HIGH, the highest value of all the samples is written; and if AVE, the average of all the samples are written.

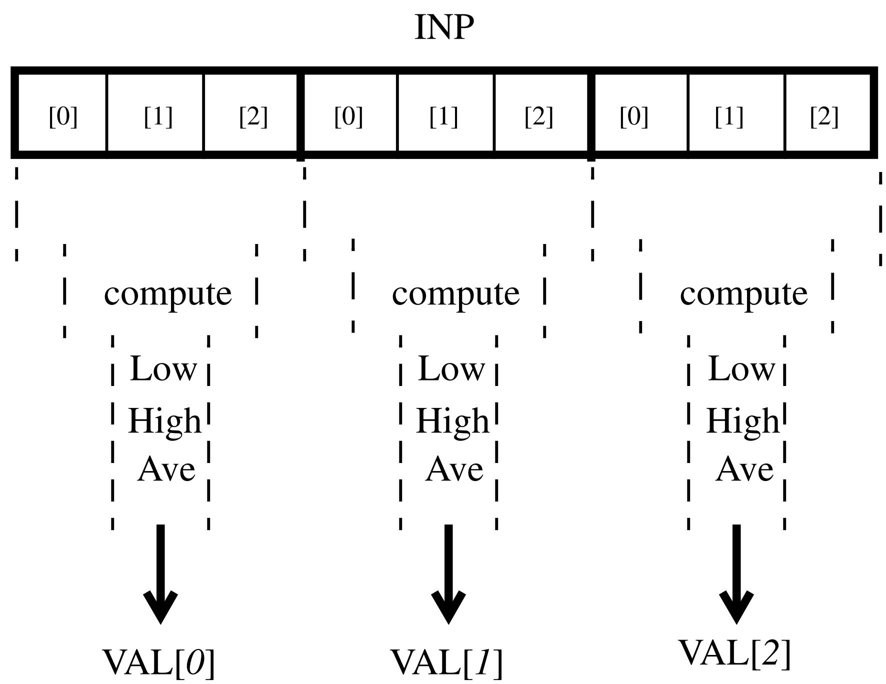
If INP refers to an array, then the following applies:

N to 1 Low Value	Compress N to 1 samples, keeping the lowest value
N to 1 High Value	Compress N to 1 samples, keeping the highest value
N to 1 Average	Compress N to 1 samples, taking the average

The compression record keeps (NSAM) data samples.

The N field determines the number of fields to compress into 1.

Thus, if NSAM was 3, and N was also equal to 3, then the algorithms would work as in the following diagram.



OFF number of samples are ignored at the beginning of the array being compressed. IHIL and ILIL are used as initial value filters when compressing arrays; that is, the compression does not begin until a value is found that is either greater than IHIL or less than ILIL

RES resets the algorithm before the maximum number of samples are reached.

4. Operator Display Parameters

These parameters are used to present meaningful data to the operator. They display the value and other parameters of the record either textually or graphically.

The HOPR and LOPR fields only specify the range for VAL, HIHI, HIGH, LOLO and LOW fields.

PREC controls the floating-point precision whenever `get_precision` is called, and the field being referenced is the VAL field (i.e., one of the values contained in the circular buffer).

The EGU field should be given a string that describes the value of VAL, but is used whenever the `get_units` record support routine is called.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	null	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

5. Alarm Parameters

The compression record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

6. Run-time Parameters.

These parameters are used by the run-time code for processing the data compression algorithm. They are not configurable by the user, though some are accessible at run-time. They can represent the current state of the waveform or of the record whose field is referenced by the INP field.

NUSE holds the number of elements currently stored in VAL.

BPTR is a pointer that refers to the buffer referenced by VAL.

The SPTR field pointer to an array that is used for array averages.

WPTR is used by the dbGetlinks routines.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NUSE	Number Used	ULONG	No	0	Yes	No	No	No
BPTR	Buffer Pointer	NOACCESS	No	0	No	No	No	
SPTR	Summing Buffer Pointer	NOACCESS	No	0	No	No	No	
WPTR	Work Buffer Pointer	NOACCESS	No	0	No	No	No	
CVB	Compress Value Buffer	DOUBLE	No	0	Yes	No	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
INX	Current Index of Circular Buffer	ULONG	No	0	Yes	No	No	No

7. Record Support Routines

init_record

Space for all necessary arrays is allocated. The addresses are stored in the appropriate fields in the record.

process

See next section.

special

This routine is called when RSET is set. It performs a reset.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

cvt_dbaddr

This is called by dbNameToAddr. It makes the dbAddr structure refer to the actual buffer holding the result.

get_array_info

Obtains values from the circular buffer referenced by VAL.

put_array_info

Writes values into the circular buffer referenced by VAL.

get_units

Retrieves EGU.

get_precision

Retrieves PREC.

get_graphic_double

Sets the upper display and lower display limits for a field. If the field is VAL, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_control_double

Sets the upper control and the lower control limits for a field. If the field is VAL, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

8. Record Processing

Routine process implements the following algorithm:

1. If INP is not a database link, check monitors and the forward link and return.
2. Get the current data referenced by INP.
3. Perform the appropriate algorithm:
    1. Average: Read N successive instances of INP and perform an element by element average. Until N instances have been obtained it just return without checking monitors or the forward link. When N instances have been obtained complete the algorithm, store the result in the VAL array, check monitors and the forward link, and return.
    2. Circular Buffer: Write the values obtained from INP into the VAL array as a circular buffer, check monitors and the forward link, and return.
    3. N to 1 xxx and INP refers to a scalar: Obtain N successive values from INP and apply the NTO1xxx algorithm to these values. Until N values are obtained monitors and forward links are not checked. When N successive values have been obtained, complete the algorithm, check monitors and the forward link, and return.
    4. N to 1 xxx and INP refers to an array: The ILIL and IHIL are honored if ILIL < IHIL. The input array is divided into subarrays of length N. The specified N to 1 xxx compression algorithm is applied to each sub-array and the result stored in the array referenced by VAL. The monitors and forward link are checked.
4. If success, set UDF to FALSE.
5. Check to see if monitors should be invoked:
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - NSEV and NSTA are reset to 0.
6. Scan forward link if necessary, set PACT FALSE, and return.

Chapter 12:CPID Control

1. Introduction

The CPID record, like the PID record, is used for making error corrections on output values in order to maintain a specified setpoint. In the CPID record, the setpoint must be retrieved from another database record. When the record is processed, it reads the controlled value and the setpoint, and then computes the error correction. Unlike the PID record, the CPID record can write this value because it has its own output link.

Another important difference is that unlike the PID record, the CPID record has four different modes: normal mode, sequencer mode, local mode, and manual mode. The normal mode causes the CPID record to function in the same way as the PID record does. The other modes significantly change how the CPID record functions. All these modes are explained in this chapter.

When in normal mode, the CPID record performs the same equation as the PID record. This equation consists of three parts: the proportional, the integral, and the derivative. Using the gain that exists for each of these terms, the user can weight each contribution according to the response of the process being controlled.

The fields in this record fall into several categories:

scan parameters

controlled variable and setpoint parameters

output, readback, and mode parameters

expression parameters

operator display parameters

alarm parameters

monitor parameters

run-time parameters

2. Scan Parameters

The CPID record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Since the CPID record supports no direct interfaces to hardware, it cannot be scanned on I/O interrupt, i.e., its SCAN field cannot specify `I/O Intr`.

However, the CPID record contains an additional field that affects its processing while in normal mode. The minimum delta time field (MDT) is a scanning field particular to the CPID record and PID records. It can be configured by the user or modified at run-time. It contains a floating point value which represents the minimum amount of time between record processing so that if the amount of time between the last time the record was processed and the current time is less than MDT, then DM and the output value (OVAL) are set to 0. If MDT is left at its default value (0), the minimum delta will be equal to one clock tick.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
MDT	Minimum Delta Time	FLOAT	Yes	0	Yes	Yes	No	No

3. Controlled Variable and Setpoint Parameters

The control variable location field (CVL) is used to obtain the value of the controlled process variable, the value read into the CVAL field. The link must be a database link. If it is not a database link an INVALID alarm is triggered when the record is processed. See *Address Specification, Chapter 1, 2*, for information on specifying links.

The setpoint value is set in the VAL field. Unlike the PID record, the setpoint value cannot be read from other records, but must be set via dbPuts. It is set equal to CVAL, the value retrieved from the CVL link, when the record is initialized.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
CVL	Controlled value location (an input link)	INLINK	Yes	0	No	No	N/A	No
VAL	Setpoint value	FLOAT	No	0	Yes	Yes	Yes	Yes

4. Expression Parameters

The PID expression calculated by the CPID record in Normal mode is the same as the PID record's expression. The discrete from of the PID expression is as follows:

$$M(n) = KP \times E(n) + KI \times Sumi\ (E(i) \times dT(n)) + KD \times \left(\frac{E(n) - E(n-1)}{dT(n)}\right) + Mr$$

where

M(n) = value of manipulated variable at *n*th instant.

KP, KI, and KD = Proportional, Integral, Derivative gains set by the user

E(n) = Error at *n*th sampling instant

SUMi = Sum from i=0 to i=n

dT(n) = Time difference between n-1 instance and nth instance

Mr = Midrange adjustment

Taking the first difference between instances yields the following equation:

$$\mathrm{delM(n)} = \mathrm{KP} \times (\mathrm{E(n)} - \mathrm{E(n\text{-}1)}) + \mathrm{KI} \times \mathrm{E(i)} \times \mathrm{dT(n)} + \mathrm{KD} \times \frac{\mathrm{E(n)} - \mathrm{E(n\text{-}1)}}{\mathrm{dT(n)}} - \frac{(\mathrm{E(n)} - \mathrm{E(n\text{-}1)})}{\mathrm{dT(n\text{-}1)}}$$

The terms KP, KI, and KD are the only terms configured by the user. These terms represent the gain for the proportional (KP), integral (KI), and the derivative (KD). A field exists for each of these terms (the KP, KI, and KD fields). KP should be configured according to the characteristics of the controlled variable; KI should be set equal to the number of times that the integral contribution repeats the proportional contribution; and KD should equal the number of minutes until the derivative contribution repeats the proportional contribution.

The PID record calculates the other terms of the expression:

E_n	Error at nth sampling instant, where the Error equals the setpoint minus the value of the controlled variable (VAL - CVAL).
delM(n)	This is the end result of the PID expression—the change in the manipulated value. The DM field holds this value.
dT(n)	This is the time difference between n and n-1, between current and last samplings.

The final result of the expression `delM(n)` and the other parts of the expression are held in several different run-time fields, which are not configurable prior to run-time, nor modifiable at run-time, but can be accessed so that their values can be used.

When in Normal Mode, the DM field contains the change in manipulated value, i.e., the result of the PID expression delM(n). For other modes, the value is different, but is always an increment which can be accessed by the desired output link (DOL) of an analog output record. Since its value represents an increment, the OIF field of the analog output record should be set to `Incremental`. Note that the final result to be written may not be the same as the final result of the equation; that is, DM, the result of the equation, may not equal OVAL, the output value, depending on the mode.

The P, I, D, CT, DT, ERR, and DERR fields are used to calculate the PID expression. They are not configurable prior to, nor modifiable during, run-time, but may be of interest when fine tuning the gains of each contribution (KP, KI, KD). The following lists the fields as they relate to the expression:

ERR	E_n. Error at current sampling (VAL-CVAL).
DERR	E_n - E_{n-1}. Difference between current error and error at last sampling.
P	This field corresponds to `KP * DERR`
I	This field corresponds to `En * dTn * KI`
D	This field corresponds to `Kp * Kd * (err/dt(n) – derr/dt(n–1))`

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
KP	Proportional Gain	FLOAT	Yes	0	Yes	Yes	No	No
KI	Integral Gain, in repeats per minute.	FLOAT	Yes	0	Yes	Yes	No	No
KD	Derivative Gain, in repeats per minute	FLOAT	Yes	0	Yes	Yes	No	No
DM	Change in Manipulated Value	FLOAT	No	0	Yes	No	Yes	No
P	Proportional contribution to DM.	FLOAT	No	0	Yes	No	Yes	No
I	Integral contribution to DM.	FLOAT	No	0	Yes	No	Yes	No
D	Derivative contribution to DM.	FLOAT	No	0	Yes	No	Yes	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
CT	Clock ticks when previous process occurred.	ULONG	No	0	Yes	No	Yes	No
DT	Time difference in seconds between processing steps.	FLOAT	No	0	Yes	No	Yes	No
ERR	Current error (VAL - CVAL).	FLOAT	No	0	Yes	No	Yes	No
DERR	Delta Error	FLOAT	No	0	Yes	No	Yes	No

5. Output, Readback, and Mode Parameters

Unlike the PID record, the CPID record has an output link so that it can write its output. The value written is the value of the OVAL field. How record processing determines this value depends on, among other things, the current mode and the limits.

Firstly, the record has two output modes, CHANGE and POSITION, set by the OMOD field. If OMOD is (0,1), then the output mode will be (CHANGE, POSITION). The OMOD field cannot be modified at run-time, so it must be configured to be either CHANGE or POSITION. It is initialized to CHANGE by default.

When CHANGE, the record sets OVAL equal to DM, the result of the PID expression. DM being itself an increment, OVAL is an incremental value in the CHANGE mode. If the record is in POSITION mode, OVAL is an absolute value obtained by adding OVAL to DM. In this case, OVAL will be the last value of OVAL when it was written unless it was changed by database access.

```
if CHANGE
      OVAL = DM
if POSITION
      OVAL = OVAL + DM
```

After OVAL is adjusted in case it violates the value limits explained below, DM is set equal to OVAL when in CHANGE mode. When in position mode, after any adjustments, DM is set equal to OVAL - MLST, MLST being the value of OVAL the last time the record was processed. This way, DM is always an increment, adjusted to be within the value limits.

Secondly, there are four different modes on top of the output mode: Normal, Sequencer, Manual, and Local.

1. In Normal mode, OVAL is computed as explained above. DM is also computed as usual.
2. In Sequencer mode, the CPID record computes OVAL as SVAL - ORBV when in CHANGE mode, and sets OVAL equal to SVAL in the POSITION mode. DM is computed as SVAL - ORBV in CHANGE mode and OVAL - SVAL in POSITION mode. The sequencer program should determine the value of SVAL while in Sequencer mode.
3. In Manual mode, the CPID record computes OVAL as MVAL - ORBV when in CHANGE mode, and sets OVAL equal to MVAL when in POSITION mode. DM is set equal to MVAL - ORBV in CHANGE mode and OVAL - MVAL in POSITION mode. MVAL is determined by the operator at run-time, via dbPuts.
4. In Local mode, the CPID record sets OVAL equal to 0 when the output mode is CHANGE and ORBV when the output mode is POSITION. The Local mode was implemented to support the setting of the output by a local hardware control.

The MMOD, SMOD, and LOC field determine which of these four modes the CPID record is in. The MMOD and SMOD fields are set via database access (by the operator or the sequencer). The LOC field must specify a database link if the record is to be placed into Local mode. If the MMOD field is true (has a value other than 0), then the mode is Manual. If the SMOD field is true, then the mode is Sequencer. If the LOC field is a database link and the value retrieved from it is 0, then the mode is Local. The default mode is the normal mode so that if none of the above conditions exist, the mode is normal. An order of precedence exists so that the mode can be determined when more than one of the above conditions is true. The order of precedence is Local mode, Manual mode, Sequencer, then Normal mode, so that if the MMOD and SMOD fields are true and the value retrieved from the LOC field *is not 0* (i.e., local mode is not requested), then the mode is Sequencer.

Note: DM is calculated even for modes other than Normal. This prevents the DT change in time contribution from “winding up” when the mode is other than Normal.

Finally, MAX, MIN, DMAX, and DMIN fields are used to specify limits for the output value. The MAX and MIN fields limit the value of OVAL, whereas the DMAX and DMIN fields limit the value of DM. How these fields do this for the different modes is somewhat complex. See Section 11, *Record Processing*, in this chapter for a further explanation of how these fields work.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OUT	Output Link	OUTLINK	Yes	0	No	No	N/A	No
ORBL	Output Readback Location	INLINK	Yes		No	No	No	No
OMOD	Output Mode	RECCHOICE	Yes	0	Yes	No	No	No
ORBV	Output Readback Value	FLOAT	No	0	Yes	Yes	No	No
MAX	Maximum Limit	FLOAT	Yes	0	Yes	Yes	No	No
MIN	Minimum Limit	FLOAT	Yes	0	Yes	Yes	No	No
DMAX	Maximum Change	FLOAT	Yes	0	Yes	Yes	No	No
DMIN	Minimum Change	FLOAT	Yes	0	Yes	Yes	No	No
LOC	Local Mode Switch	INLINK	Yes			No	No	No
MMOD	Manual Mode Request	GBLCHOICE	Yes	0	Yes	Yes	No	No
SMOD	Sequencer Mode Request	GBLCHOICE	Yes	0	Yes	Yes	No	No
MVAL	Manual Value	FLOAT	Yes	0	Yes	Yes	No	No
SVAL	Sequencer Value	FLOAT	Yes	0	Yes	Yes	No	No

6. Operator Display Parameters

These parameters are used to present meaningful data to the operator. These fields are used to display the setpoint (VAL), the controlled variable (CVAL), the change in manipulated value (DM), and other fields of the PID, either textually or graphically.

EGU is a string of up to 16 characters describing the units of PID’s manipulated values measures. It is retrieved by the `get_units` record support routine. It must be configured by the user if at all.

The HOPR and LOPR fields set the upper and lower display limits for the VAL, HIHI, HIGH, LOW, LOLO, VAL, P, I, D, and CVAL fields.

The PREC field determines the floating point precision with which to display VAL and CVAL. It is used whenever the `get_precision` record support routine is called.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING	Yes	null	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING	Yes	0	Yes	No	No	
DESC	Description	STRING	Yes	Null	Yes	Yes	No	No

7. Alarm Parameters

The possible alarm conditions for PID are the SCAN alarm, limit alarms, and an INVALID alarm that is triggered when CVL is not a database link. The SCAN and INVALID alarms are called by the record routines and are always of MAJOR severity.

The limit alarms trigger alarms on the VAL field. They are configured by the user in the HIHI, LOLO, HIGH, and LOW fields using floating point values. For each of these fields, there is a corresponding severity field which can be either NO ALARM, MINOR, or MAJOR. See *Alarm Specification, Chapter 1, 4*, for a complete explanation of alarms and these fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HIHI	Hihi Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HIGH	High Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOW	Low Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOLO	Lolo Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HHSV	Hihi Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HSV	High Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LSV	Low Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LLSV	Lolo Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HYST	Alarm Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

8. Monitor Parameters

These parameters are deadbands configured by the user which determine when to send monitors on the DM field. If the value of DM is greater than ADEL, archive monitors for the OVAL, CT, and DM fields are sent. Additionally, if the record is in Normal mode, monitors for P, I, D, ERR, DERR, and DT are sent. MDEL is used the same way as ADEL for all other types of monitors.

The ODEL field is not implemented in the PID record.

See *Monitor Specification, Chapter 1, 5*, for a complete explanation of monitors.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ADEL	Archive Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
MDEL	Monitor, i.e. value change, Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
ODEL	Output deadband for DM	FLOAT	Yes	0	Yes	Yes	No	No

9. Run-time Parameters

These parameters are used by the run-time code for processing the PID. They are not configurable prior to run-time. They represent the current state of the PID and/or the terms of the PID expression. Many of them are used to process the PID record more efficiently.

The CVAL field is the controlled variable value retrieved from the CVL link.

The LALM field is used to implement the hysteresis factor for the alarms.

The ALST field holds the last value for OVAL, but has no other function for the record.

The MLST field holds the last value for the OVAL field. It is used to calculate a value for DM when the record is in the POSITION mode. After any other calculations, DM is set equal to OVAL - MLST. Thus, even in POSTION mode, DM always holds a relative value, the difference between DM and the value of OVAL from the last time the record was processed.

The PMOD field indicates the current mode of the record:

0 Normal Mode

1 Manual Mode

2 Sequencer Mode

3 Local Control Mode

The last mode field (LMOD) implements monitors for PMOD. If PMOD is not equal to LMOD, monitors are posted for the PMOD field.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
CVAL	Value of controlled variable	FLOAT	No	0	Yes	Yes/No	No	No
OVAL	Output Value	FLOAT	No	0	Yes	Yes	Yes	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ODM	Old DM.	FLOAT	No	0	Yes	No	Yes	No
LALM	Value from when last monitors for alarm were triggered	FLOAT	No	0	Yes	No	No	No
ALST	Value when last monitors for archiver were triggered	FLOAT	No	0	Yes	No	No	No
MLST	Value when last monitors for value changes were triggered	FLOAT	No	0	Yes	No	No	No
LOVL	Last Readback Value	FLOAT	No	0	Yes	No	No	No
PMOD	Current Mode	RECCHOICE	Yes	0		No	Yes	No
LMOD	Last Mode	RECCHOICE	Yes	0	yes	No	No	No
ODM	Old DM Value	FLOAT	No	0	Yes	No	No	No

10. Record Support Routines

init_record

Set UDF to false. Return 0.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_units

Retrieves EGU.

get_precision

Retrieves PREC if field is CVAL or VAL. Otherwise, calls **recGblGetPrec**().

get_graphic_double

Sets the following values if field is VAL, HIHI, HIGH, LOW, LOLO, P, I, D, or CVAL:

upper_disp_limit = hopr

lower_disp_limit = lopr

get_control_double

Sets the following values if field is VAL, HIHI, HIGH, LOW, LOLO, P, I, D, or CVAL:

upper_ctrl_limit = hopr

lower_ctrl_limit = lopr

get_alarm_double

Sets the following values:

upper_alarm_limit = hihi

upper_warning_limit = high

lower_warning_limit = low

lower_alarm_limit = lolo

11. Record Processing

Routine process implements the following algorithm:

1. Set PACT to TRUE.
2. If CVL is not a database link an INVALID alarm is declared, PACT is set to FALSE, and the routine returns (0).
3. The current value of CVAL is obtained from CVL.
4. If ORBL is a database link, the current value of ORBV is obtained.
5. Determine the mode:
   mode = normal
   if SMOD true
       mode = Sequencer mode
   if MMOD true
       mode = Manual mode
   if value retrieved from LOC is 0
       mode = Local Mode
6. Compute time difference between current and last processing; that is , compute DT.
7. Set DM = 0.
8. If Mode is Normal and DT < MDT, calculate DM using PID expression.
9. If Mode is Normal and OMOD is CHANGE:
   if (ORBV + DM) is less than MAX
       set OVAL = MAX - ORBV
   else if (ORBV + DM) < MIN
       set OVAL = MIN - ORBV
   else set OVAL = DM
10. If Mode is Normal and OMOD is POSITION and OVAL > MAX:

if (OVAL - MAX) > DMAX
    set OVAL = OVAL - DMAX
if (OVAL - MAX) >= DMIN
    set OVAL = MAX

11. If Mode is Normal and OMOD is POSITION and OVAL < MIN.
    if (MIN - OVAL) > DMAX
        Set OVAL = OVAL + DMAX
    if (MIN - OVAL) >= DMIN
        set OVAL = MIN
12. If Mode is Manual and OMOD is CHANGE, set DM and OVAL to MVAL - ORBV.
13. If Mode is Manual and OMOD is POSITION, set DM and OVAL to MVAL - ORBV.
14. if Mode is Local Control and OMOD is CHANGE, set OVAL equal to 0.
15. If Mode is Local Control and OMOD is POSITION, set OVAL equal to ORBV.
16. If Mode is Sequencer and OMOD is CHANGE, set DM and OVAL to SVAL - ORBV
17. If Mode is Sequencer and OMOD is POSITION, set DM and OVAL to SVAL.
18. If OMOD is CHANGE, set DM equal to OVAL.
19. If OMOD is POSITION, set DM equal to OVAL - MLST.
20. Write OVAL to output location.
21. Check alarms and post if any.
22. Check monitors and post if any.
23. Process FLNK.
24. Set PACT equal to FALSE and return 0.

12. Device Support

The CPID record has no associated device support.

Chapter 13:dfanout

Johnny Tang, Matthew Bickley, and Chip Watson
Continuous Electron Beam Accelerator Facility
Southeastern Universities Research Association

1. Introduction

The dfanout record or data fanout record is used to forward data to up to eight other records. It's similar to the fanout record except that the capability to forward data has been added to it. It has no associated device support. The fields in this record can be classified into the following categories:

- scan parameters
- desired output parameters
- write parameters
- operator display parameters
- alarm parameters
- monitor parameters
- run-time and simulation mode parameters

2. Scan Parameters

The data fanout record has the standard fields for specifying under what circumstances it will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Since the data fanout record supports no direct interfaces to hardware, it cannot be scanned on I/O interrupt, so its SCAN field cannot be `I/O Intr`.

3. Desired Output Parameters

The data fanout record must specify where the desired output value originates, i.e., the data which it is to forward to the records in its output links. The output mode select (OMSL) field determines whether the output originates from another record or from run-time database access. When set to `closed loop`, the desired output is retrieved from the link specified in the desired output (DOL) field, which can specify either a database or channel access link, and placed into the VAL field. When set to `supervisory`, the desired output can be written to the VAL field via dpPuts at run-time.

The DOL field can also be a constant in which case the VAL field is initialized to the constant value.

Note that there are no conversion parameters, so the desired output value undergoes no conversions before it is sent out to the output links.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
DOL	Desired Output Location (Input Link)	INLINK	Yes	0	No	No	N/A	No
OMSL	Output Mode Select	GBLCHOICE	Yes	0	Yes	Yes	No	No
VAL	Value Field	DOUBLE	No	0	Yes	Yes	No	Yes

4. Write Parameters

The OUTA-OUTH fields specify where VAL is to be sent. Each field that is to forward data must specify an address to another record. See *Address Specification, Chapter 1, 2*, for information on specifying links.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OUTA	Output Link A	OUTLINK	Yes	0	No	No	N/A	No
OUTB	Output Link B	OUTLINK	Yes	0	No	No	N/A	No
OUTC	Output Link C	OUTLINK	Yes	0	No	No	N/A	No
OUTD	Output Link D	OUTLINK	Yes	0	No	No	N/A	No
OUTE	Output Link E	OUTLINK	Yes	0	No	No	N/A	No
OUTF	Output Link F	OUTLINK	Yes	0	No	No	N/A	No
OUTG	Output Link G	OUTLINK	Yes	0	No	No	N/A	No
OUTH	Output Link H	OUTLINK	Yes	0	No	No	N/A	No

5. Operator Display Parameters

These parameters are used to present meaningful data to the operator. They display the value and other parameters of the data fanout record either textually or graphically.

The EGU field can contain a string of up to 16 characters describing the value in the VAL field.

The HOPR and LOPR fields determine the upper and lower display limits for graphics displays and the upper and lower control limits for control displays. They apply to the VAL, HIHI, HIGH, LOW, and LOLO fields. The record support routines `get_graphic_double` or `get_control_double` retrieve HOPR and LOPR

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	null	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

The possible alarm conditions for data fanouts are the SCAN, READ, INVALID, and limit alarms. The SCAN and READ alarms are called by the record routines. The limit alarms are configured by the user in the HIHI, LOLO, HIGH, and LOW fields using floating-point values. The limit alarms apply only to the VAL field. The severity for each of these limits is specified in the corresponding field (HHSV, LLSV, HSV, LSV) and can be either NO_ALARM, MINOR, or MAJOR. In the hysteresis field (HYST) can be entered a number which serves as the deadband on the limit alarms.

See *Alarm Specification, Chapter 1, 4*, for a complete explanation of alarms and these fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types. See *Invalid Alarm Output Action, Chapter 3, 3.5* for more information about the IVOA and IVOV fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HIHI	Hihi Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HIGH	High Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOW	Low Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOLO	Lolo Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HHSV	Hihi Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HSV	High Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LSV	Low Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LLSV	Lolo Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HYST	Alarm Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
IVOA	Invalid Alarm Output Action	GBLCHOICE	Yes	0	Yes	Yes	No	No
IVOV	Invalid Alarm Output Value	DOUBLE	Yes	0	Yes	Yes	No	No

7. Monitor Parameters

These parameters are used to determine when to send monitors placed on the VAL field. The monitors are sent when the value field exceeds the last monitored field by the specified deadband, ADEL for archiver monitors and MDEL for all other types of monitors. If these fields have a value of zero, everytime the value changes, a monitor will be triggered; if they have a value of -1, everytime the record is scanned, monitors are triggered. See *Monitor Specification, Chapter 1, 5*, for a complete explanation of monitors.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ADEL	Archive Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
MDEL	Monitor, i.e. value change, Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

8. Run-Time Parameters and Simulation Mode Parameters

These parameters are used by the run-time code for processing the data fanout record. They are not configurable. They are used to implement the hysteresis factors for monitor callbacks.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LALM	Last Alarm Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No
ALST	Last Archiver Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No
MLST	Last Value Change Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No

9. Record Support Routines

init_record()

This routine initializes all output links that are defined. Then it initializes DOL is DOL is a constant or PV_LINK. When initializing the output links and the DOL link, a non-zero value is returned if an error occurs.

process()

See next section.

get_value()

This routine fills in the members of `struct valueDes{}` with the VAL fields value and characteristics.

get_units()

The routine copies the string specified in the EGU field to the location specified by a pointer which is passed to the routine.

get_graphic_double()

If the referenced field is VAL, HIHI, HIGH, LOW, or LOLO, this routine sets the `upper_disp_limit` member of the `dbr_grDouble{}` structure to HOPR and the `lower_disp_limit` member to LOPR. If the referenced field is not one of the above fields, then the `recGblGetControlDouble()` routine is called.

get_control_double()

Same as the `get_graphic_double` routine except that it uses the `dbr_ctrlDouble{}` structure.

get_alarm_double()

This sets the members of the `dbr_alDouble{}` structure to the specified alarm limits if the referenced field is VAL:

```
upper_alarm_limit = HIHI
upper_warning_limit = HIGHT
lower_warning_limit = LOW
lower_alarm_limit = LOLO
```

If the referenced field is not VAL, the `recGblGetAlarmDouble()` routine is called.

10. Record Processing

1. The `process()` routine first retrieves a value for DOL and places it in VAL if OMSL is set to closed loop mode. If ann error occurs, then UDF is set to FALSE.
2. PACT is set TRUE.
3. VAL is then sent to all the records specified in the OUTA-OUTH fields by calling the `recGblPutLinkValue()` for each link.
4. Alarms are checked and monitors are called if conditions apply.
5. The data fanout's own forward link is then processed.
6. PACT is set FALSE, and the `process()` routine returns. A -1 is returned if there was an error writing values to one of the output links.

Chapter 14:Event

1. Introduction

The normal use for this record type is to post an event and/or process a forward link. Device support for this record can provide a hardware interrupt handler routine for I/O Event-scanned records. The records in this field fall into the following groups of parameters:

scan parameters

read parameters

event number parameters

simulation mode parameters

2. Scan Parameters

The event record has the standard fields for specifying under what circumstances it will be processed. If the SCAN field specifies `I/O Intr`, then device support will provide an interrupt handler, posting an event number when an I/O interrupt occurs. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how the scanning fields work. Note that I/O event scanning is only supported for those card types that interrupt.

3. Input Specification

The device support routines use the address in this record to obtain input. For records that provide an interrupt handler, the INP field should specify the address of the I/O card, and the DTYP field should specify a valid device support module. Be aware that the address format differs according to the card type used. See *Address Specification, Chapter 1, 2*, for information on the format of hardware addresses and specifying links. You can see a list of the device support modules currently supported at the user's local site by using the dbst utility in R3.13.

For soft records, the INP field can be a constant, a database link, or a channel access link. For soft records, the DTYP field should specify `Soft Channel`.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
INP	Input Link	INLINK	Yes	0	No	No	N/A	No
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	

4. Event Number Parameters

The VAL field contains the event number read by the device support routines. It is this number which is posted. For records that use `Soft Channel` device support, it can be configured before run-time or set via dbPuts.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Event Number to Post	USHORT	Yes	0	Yes	Yes	Yes	No

5. Operator Display Parameters

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

The Event record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

7. Simulation Mode Parameters

The following fields are used to operate the event record in the simulation mode. See Chapter 3, *Fields Common to Many Record Types*, for more information on these fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No
SVAL	Simulation Value	DOUBLE	No	0	Yes	Yes	No	No
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

8. Record Support Routines

init_record

This routine initializes SIMM with the value of SIML if SIML type is a CONSTANT link or creates a channel access link if SIML type is PV_LINK. SVAL is likewise initialized if SIOL is CONSTANT or PV_LINK.

If device support includes init_record, it is called.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

9. Record Processing

Routine process implements the following algorithm:

1. readValue is called. See *Input Records, Chapter 3, 2*, for more information.
2. If PACT has been changed to TRUE, the device support read routine has started but has not completed reading a new input value. In this case, the processing routine merely returns, leaving PACT TRUE.
3. If VAL > 0, post event number VAL.

4. Check to see if monitors should be invoked. Alarm monitors are invoked if the alarm status or severity has changed NSEV and NSTA are reset to 0.
5. Scan forward link if necessary, set PACT FALSE, and return.

10. Device Support

10.1. Fields of Interest To Device Support

Each record must have an associated set of device support routines. The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See Chapter 2, *Fields Common to All Record Types*, for an explanation of these fields.
DPVT	Device Private	
UDF	VAL Undefined	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
INP	Input Link	This field is used by the device support routines to locate its input.
PRIO	Priority	This value must be used by the device support interrupt handler to set the scheduling priority for processing this record.

10.2. Device Support Routines

Device support consists of the following routines:

report

```
report(FILE fp, interest)
```

Not currently used.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
    *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

read_event

```
read_event(precord)
```

This routine returns the following values:

- •0: Success.
- •Other: Error.

10.3. Device Support For Soft Records

The `Soft Channel` device support module is available. The INP link type must be either CONSTANT, DB_LINK, or CA_LINK.

If the INP link type is CONSTANT, then the constant value is stored into VAL by init_record, and UDF is set to FALSE. If the INP link type is PV_LINK, then dbCaAddInlink is called by init_record.

`read_event` calls recGblGetLinkValue to read the current value of VAL. See *Input Records, Chapter 3, 2*, for details on soft input.

Chapter 15:Fanout

1. Introduction

The fanout record uses several forward processing links to force multiple passive records to scan. When more than one record needs to be scanned as the result of a record being processed, the forward link of that record can specify a fanout record. The fanout record can specify up to six other records to process. When more than six are needed, one of the links in the fanout record can point to another fanout record.

> **NOTE: Fanout records only propagate processing, not data. The dfanout or data fanout record can, on the other hand, send data to up to eight other records.**

The fanout record's fields fall into the following categories:

scan parameters

operator display parameters

run-time parameters.

2. Scan Parameters

The forward link fields of the fanout record (LINK1-LINK6) specify records to be scanned. The records to be processed must specify `Passive` in their SCAN fields; otherwise the forward link to it will not cause it to process. Also when specifying database links for the fanout record, the user needs only to specify the record name. As no value is being sent or retrieved, the field name is optional.

The SELM, SELN, and SELL fields specify the order of processing for the forward links. The select mechanism field (SELM) has three choices—`All`, `Specified`, or `Mask`. How these affect which links to process and in which order is as follows:

All	Links are processed in numerical order—LNK1, LNK2, etc.
Specified	SELN is used as the specifier of which link to process. For instance, if SELN=1, then LNK1 will be processed.
Mask	The number in SELN is used as follows: if SELN=1, LNK1 is processed. If SELN=2, LNK1 and LNK2 are processed. If SELN=3, then LNK1, LNK2, and LNK3 are processed, etc.

SELN reads its value from SELL. SELL can be a constant, a database link, or a channel access link. If a constant, SELN is initialized with the constant value and can be changed via dbPuts. For database/channel access links, SELN is retrieved from SELL each time the record is processed and can also be changed via dbPuts.

The Fanout record also has the standard scanning fields common to all records. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains in more detail how forward links and the scanning algorithms work.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LNK1	Forward Links	FWDLINK	Yes	0	No	No	N/A	No
LNK2	Forward Links	FWDLINK	Yes	0	No	No	N/A	No
LNK3	Forward Links	FWDLINK	Yes	0	No	No	N/A	No
LNK4	Forward Links	FWDLINK	Yes	0	No	No	N/A	No
LNK5	Forward Links	FWDLINK	Yes	0	No	No	N/A	No
LNK6	Forward Links	FWDLINK	Yes	0	No	No	N/A	No
SELM	Select Mechanism:	RECCHOICE	Yes	0	Yes	Yes	No	No
SELN	Link Selection Algorithm	USHORT	No	1	Yes	Yes	No	No
SELL	Link Selection Location	INLINK	Yes	0	No	No	N/A	No

3. Operator Display Parameters

These parameters are used to present meaningful data to the operator. See Chapter 2, *Fields Common to All Record Types*, for more on these fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

4. Alarm Parameters

The Fanout record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

5. Run-time Parameters

The VAL field is used only so that dbNameAddr succeeds when no field name is specified. Otherwise, it has no significance.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Value Field	LONG	No	0	Yes	Yes	No	Yes

6. Record Support Routines

init_record

This routine initializes SELN with the value of SELL, if SELL type is CONSTANT link, or creates a channel access link if SELL type is PV_LINK.

process

See next section.

7. Record Processing

Routine process implements the following algorithm:

1. PACT is set to TRUE.
2. The link selection SELN is fetched.
3. Depending on the selection mechanism, the link selection forward links are processed. and UDF is set to FALSE.
4. Check to see if monitors should be invoked.
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - NSEV and NSTA are reset to 0.
5. Scan forward link if necessary, set PACT FALSE, and return.

Chapter 16:Histogram

1. Introduction

The histogram record is used to store frequency counts of a signal into an array of arbitrary length. The user can configure the range of the signal value that the array will store. Anything outside this range will be ignored. The fields in this record fall into the following categories:

scan parameters

read parameters

operator display parameters

monitor parameters

run-time and simulation mode parameters

2. Scanning Parameters

The histogram record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used.

3. Read Parameters

The SVL is the input link where the record reads its value. It can be a constant, a database link, or a channel access link. If SVL is a database or channel access link, then SGNL is read from SVL. If SVL is a constant, then SGNL is initialized with the constant value but can be changed via dbPuts. The `Soft Channel` device support module can be specified in the DTYP field. For a list of other device support modules currently supported at the user's local site, use the `dbst` utility in R3.13.

The ULIM and LLIM fields determine the usable range of signal values. Any value of SGNL below LLIM or above ULIM is outside the range and will not be stored in the array. In the NELM field the user must specify the array size, e.g., the number of array elements. Each element in the NELM field holds the counts for an interval of the range of signal counts, the range specified by ULIM and LLIM. These intervals are determined by dividing the range by NELM:

```
(ULIM - LLIM) / NELM.
```

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SVL	Signal Value Location (an input link)	INLINK	Yes	0	No	No	N/A	No
SGNL	Signal Value	DOUBLE	No	0	Yes	Yes	Yes	No
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	
NELM	Number of elements in array	USHORT	Yes	1	Yes	No	No	No
ULIM	Upper Signal Limit	DOUBLE	Yes	0	Yes	Yes	No	No
LLIM	Lower Signal Limit	DOUBLE	Yes	0	Yes	Yes	No	No

4. Operator Display Parameters

These parameters are used to present meaningful data to the operator. These fields are used to display the value and other parameters of the histogram either textually or graphically. See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

5. Alarm Parameters

The Histogram record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

6. Monitor Parameters

The MDEL field implements the monitor count deadband. Only when MCNT is greater than the value given to MDEL are monitors triggered, MCNT being the number of counts since the last time the record was processed. If MDEL is -1, everytime the record is processed, a monitor is triggered regardless.

If SDEL is greater than 0, it causes a callback routine to be called. The number specified in SDEL is the callback routines interval. The callback routine is called every SDEL seconds. The callback routine posts an event if MCNT is greater than 0.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
MDEL	Monitor Delta	SHORT	Yes	0	Yes	Yes	No	No
SDEL	Monitor Seconds Deadband	FLOAT	Yes	0	Yes	No	No	No

7. Run-time and Simulation Mode Parameters

These parameters are used by the run-time code for processing the histogram. They are not configurable by the user prior to run-time. They represent the current state of the record. Many of them are used to process the histogram more efficiently.

The BPTR field contains a pointer to the unsigned long array of frequency values. The VAL field references this array as well. However, the BPTR field is not accessible at run-time.

The MCNT field keeps counts the number of signal counts since the last monitor was invoked.

The WDOG field contains a pointer to the callback control structure and is of no interest to the user.

The collections controls field (CMD) is a menu field with five choices: `Read`, `Clear`, `Start`, `Stop`, and `Setup`. When `Read`, the record retrieves its values and adds them to the signal array. This command will first clear the signal counts which have already been read when it is first invoked. The `Clear` command erases the signal counts, setting the elements in the array back to zero. Afterwards, the field is set back to `Read`. The `Start` command simply causes the record to read signal values into the array. Unlike `Read`, it doesn't clear the array first. The `Stop` command disables the reading of signal values into the array. The `Setup` command waits until the `start` or `read` command has been issued to start counting.

The CSTA or collections status field implements the CMD field choices by enabling or disabling the reading of values into the histogram array. While FALSE, no signals are added to the array. While TRUE, signals are read and added to the array. The field is initialized to TRUE. The `Stop` command is the only command that sets CSTA to FALSE. On the other hand, the `Start` command is the only command that sets it to TRUE. Thus, `Start` must be invoked after each `Stop` command in order to enable counting; invoking `Read` will not enable signal counting after `Stop` has been invoked.

A typical use of these fields would be to initialize the CMD field to `Read` (it is initialized to this command by default), to use the `Stop` command to disable counting when necessary, after which the `Start` command can be invoked to re-start the signal count.

The WDTH field is a private field that holds the signal width of the array elements. For instance, if the LLIM was configured to be 4.0 and ULIM was configured to be 12.0 and the NELM was set to 4, then the WDTH for each array would be 2. Thus, it is (ULIM - LLIM) / NELM.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
BPTR	Buffer Pointer	NOACCESS	No	0	No	No	No	No
VAL	Value Field	See BPTR	No	0	Yes	No	No	No
MCNT	Monitor Counts	SHORT	No	0	Yes	No	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
WDOG	Watchdog Callback	NOACCESS	No	0	No	No	No	No
CMD	Collections Control	RECCHOICE	No	0	Yes	Yes	No	No
CSTA	Collections Status	SHORT	No	1	Yes	No	No	No
WDTH	Element Width	DOUBLE	No	0	Yes	No	No	No

The following fields are used to operate the histogram record in simulation mode. See Chapter 3, *Fields Common to Many Record Types*, for more information on the simulation mode fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No
SVAL	Simulation Value	DOUBLE	No	0	Yes	Yes	No	No
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

8. Record Support Routines

init_record

Using NELM, space for the unsigned long array is allocated and the width WDTH of the array is calculated.

This routine initializes SIMM with the value of SIML if SIML type is CONSTANT link or creates a channel access link if SIML type is PV_LINK. SVAL is likewise initialized if SIOL is CONSTANT or PV_LINK.

This routine next checks to see that device support and a device support read routine are available. If device support includes init_record, it is called.

process

See next section.

special

Special is invoked whenever the fields CMD, SGNL, ULIM, or LLIM are changed.

If SGNL is changed, add_count is called.

If ULIM or LLIM are changed, WDTH is recalculated and clear_histogram is called.

If CMD is less or equal to 1, clear_histogram is called and CMD is reset to 0. If CMD is 2, CSTA is set to TRUE and CMD is reset to 0. If CMD is 3, CSTA is set to FALSE and CMD is reset to 0.

clear_histogram zeros out the histogram array. add_count increments the frequency in the histogram array.

get_value

Fills in the values of struct valueDes so that they refer to the array.

cvt_dbaddr

This is called by dbNameToAddr. It makes the dbAddr structure refer to the actual buffer holding the array.

get_array_info

Obtains values from the array referenced by VAL.

put_array_info

Writes values into the array referenced by VAL.

9. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. readValue is called. See *Input Records, Chapter 3, 2* for more information
3. If PACT has been changed to TRUE, the device support read routine has started but has not completed writing the new value. In this case, the processing routine merely returns, leaving PACT TRUE.
4. Add count to histogram array.
5. Check to see if monitors should be invoked. Alarm monitors are invoked if the alarm status or severity has changed. Archive and value change monitors are invoked if MDEL conditions are met. NSEV and NSTA are reset to 0.
6. Scan forward link if necessary, set PACT and INIT to FALSE, and return.

10. Device Support

10.1. Fields Of Interest To Device Support

The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See Chapter 2, *Fields Common to All Record Types* for an explanation of these fields.
DPVT	Device Private	
UDF	VAL Undefined	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
SVL	Signal Value	The device support module retrieves a value from SGNL for SVL
SGNL	Signal Value Location	

10.2. Device Support Routines

Device support consists of the following routines:

init_record

```
init_record(precord)
```

This routine is called by the record support init_record routine. It makes sure that SGNL is a CONSTANT, PV_LINK, DB_LINK, or CA_LINK. It also retrieves a value for SVL from SGNL. If SGNL is none of the above, an error is generated.

read_histogram

```
read_histogram(*precord)
```

This routine is called by the record support routines. It retrieves a value for SVL from SGNL.

10.3. Device Support For Soft Records

Only the device support module `Soft Channel` is currently provided, though other device support modules may be provided at the user's site.

Soft Channel

The `Soft Channel` device support routine retrieves a value from SGNL. SGNL must be CONSTANT, PV_LINK, DB_LINK, or CA_LINK.

Chapter 17: longin—Long Input

1. Introduction

The normal use for the long input record or "longin" record is to retrieve a long integer value of up to 32 bits. Device support routines are provided to support direct interfaces to hardware. In addition, the `Soft Channel` device module is provided to obtain input via database or channel access links or via dbPutField or dbPutLink requests.

The fields in this record fall into the following categories:

scan parameters

read parameters

operator display parameters

alarm parameters

run-time and simulation mode parameters

2. Scan Parameters

The long input record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Read Parameters

The device support routines use the INP field to obtain the record's input. For records that obtain their input from devices, the INP field must contain the address of the I/O card, and the DTYP field must specify the proper device support module. Be aware that the address format differs according to the I/O bus used. You can see a list of the device support modules currently supported at the user's local site by using the `dbst` utility in R3.13.

For soft records, the INP can be a constant, a database link, or a channel access link. The value is read directly into VAL. The `Soft Channel` device support module is available for longin records. See *Address Specification, Chapter 1, 2*, for information on the format of hardware addresses and a database links.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Value Field	LONG	No	0	Yes	Yes	Yes	Yes
INP	Input Link	INLINK	Yes	0	No	No	N/A	No
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	

4. Operator Display Parameters

These parameters are used to present meaningful data to the operator. These fields are used to display the value and other parameters of the long input either textually or graphically.

EGU is a string of up to 16 characters describing the units that the long input measures. It is retrieved by the `get_units` record support routine.

The HOPR and LOPR fields set the upper and lower display limits for the VAL, HIHI, HIGH, LOW, and LOLO fields. Both the `get_graphic_double` and `get_control_double` record support routines retrieve these fields.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	null	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

5. Alarm Parameters

The possible alarm conditions for long inputs are the SCAN, READ, and limit alarms. The SCAN and READ alarms are called by the record or device support routines.

The limit alarms are configured by the user in the HIHI, LOLO, HIGH, and LOW fields using numerical values. For each of these fields, there is a corresponding severity field which can be either NO_ALARM, MINOR, or MAJOR. The HYST field can be used to specify a deadband around each limit. See *Alarm Specification, Chapter 1, 4*, for a complete explanation of alarms and these fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HIHI	Hihi Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HIGH	High Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOW	Low Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOLO	Lolo Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HHSV	Hihi Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HSV	High Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LSV	Low Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LLSV	Lolo Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HYST	Alarm Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

6. Monitor Parameters

These parameters are used to determine when to send monitors placed on the value field. The monitors are sent when the value field exceeds the last monitored field (see the next section) by the appropriate deadband. If these fields have a value of zero, everytime the value changes, a monitor will be triggered; if they have a value of -1, everytime the record is scanned, monitors are triggered. The ADEL field is used by archive monitors and the MDEL field for all other types of monitors. See *Monitor Specification, Chapter 1, 5*, for a complete explanation of monitors.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ADEL	Archive Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
MDEL	Monitor, i.e. value change, Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

7. Run-time and Simulation Mode Parameters

The LALM, MLST, and ALST fields are used to implement the hysteresis factors for monitor callbacks. Only if the difference between these fields and the corresponding value field is greater than the appropriate delta (MDEL, ADEL, HYST)—only then are monitors triggered. For instance, only if the difference between VAL and MLST is greater than MDEL are the monitors triggered for VAL.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LALM	Last Alarm Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No
ALST	Last Archiver Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No
MLST	Last Value Change Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No

The following fields are used to operate the long input in the simulation mode. See Chapter 3, *Fields Common to Many Record Types*, for more information on these fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No
SVAL	Simulation Value	DOUBLE	No	0	Yes	Yes	No	No
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

8. Record Support Routines

init_record

This routine initializes SIMM with the value of SIML if SIML type is CONSTANT link or creates a channel access link if SIML type is PV_LINK. SVAL is likewise initialized if SIOL is CONSTANT or PV_LINK.

This routine next checks to see that device support is available and a device support read routine is defined. If either does not exist, an error message is issued and processing is terminated.

If device support includes init_record, it is called.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_units

Retrieves EGU.

get_graphic_double

Sets the upper display and lower display limits for a field. If the field is VAL, HIHI, HIGH, LOW, or LOLO, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_control_double

Sets the upper control and the lower control limits for a field. If the field is VAL, HIHI, HIGH, LOW, or LOLO, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_alarm_double

Sets the following values:

upper_alarm_limit = HIHI

upper_warning_limit = HIGH

lower_warning_limit = LOW

lower_alarm_limit = LOLO

9. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. readValue is called. See *Input Records, Chapter 3, 2* for more information.
3. If PACT has been changed to TRUE, the device support read routine has started but has not completed reading a new input value. In this case, the processing routine merely returns, leaving PACT TRUE.

4. Check alarms. This routine checks to see if the new VAL causes the alarm status and severity to change. If so, NSEV, NSTA and LALM are set. It also honors the alarm hysteresis factor (HYST). Thus the value must change by more than HYST before the alarm status and severity is lowered.
5. Check to see if monitors should be invoked.
   - Alarm monitors are invoked if the alarm status or severity has changed.
   - Archive and value change monitors are invoked if ADEL and MDEL conditions are met.
   - NSEV and NSTA are reset to 0.
6. Scan forward link if necessary, set PACT FALSE, and return.

10. Device Support

10.1. Fields Of Interest To Device Support

Each long input record must have an associated set of device support routines. The primary responsibility of the device support routines is to obtain a new input value whenever read_longin is called. The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See Chapter 2, *Fields Common to All Record Types* for an explanation of these fields.
DPVT	Device Private	
UDF	VAL Undefined	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
VAL	Value Field	This field is set by device support routines.
INP	Input Link	This field is used by the device support routines to locate its input.

10.2. Device Support routines

Device support consists of the following routines:

report

```
report(FILE fp, paddr)
```

Not currently used.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
    *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

read_longin

```
read_longin(precord)
```

This routine must provide a new input value. It returns the following values:

- 0: Success. A new value is placed in VAL.
- Other: Error.

10.3. Device Support For Soft Records

The `Soft Channel` device support module places a value directly in VAL.

If the INP link type is constant, then the constant value is stored into VAL by init_record, and UDF is set to FALSE. If the INP link type is PV_LINK, then dbCaAddInlink is called by init_record.

`read_longin` calls recGblGetLinkValue to read the current value of VAL. See *Soft Input, Chapter 3, 2.3* for more information

If the return status of `recGblGetLinkValue` is zero then read_longin sets UDF to FALSE. read_longin returns the status of `recGblGetLinkValue`.

Chapter 18:longout - Long Output

1. Introduction

The normal use for the long output or "longout" record type is to store long integer values of up to 32 bits and write them to hardware devices. The `Soft Channel` device support routine can also be used to write values to other records via database or channel access links. The OUT field determines how the record is used. The record supports alarm limits and graphics and control limits.

The fields in this record fall into the following categories:

scan parameters

desired output parameters

write parameters

operator display parameters

alarm parameters

monitor parameters

simulation mode parameters

2. Scan Parameters

The longout record has the standard fields for specifying under what circumstances it will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Desired Output Parameters

The data fanout record must specify where the desired output originates, i.e., the 32 bit integer value it is to write. The output mode select (OMSL) field determines whether the output originates from another record or from database access. When set to `closed_loop`, the desired output is retrieved from the link specified in the desired output (DOL) field—which can specify either a database or channel access link—and placed into the VAL field. When set to `supervisory`, the desired output can be written into the VAL field via dpPuts at run-time.

The DOL field can be configured as constant, so that when the record is initialized, the VAL field will be initialized with this VALUE. If DOL is a constant, then the OMSL loop cannot be `closed_loop`.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor
DOL	Desired Output Location (input link)	INLINK	Yes	0	No	No	N/A
OMSL	Output Mode Select	GBLCHOICE	Yes	0	Yes	Yes	No
VAL	Value Field	LONG	No	0	Yes	Yes	Yes

4. Write Parameters

The OUT link field determines where the record is to send its output. For records that write values to hardware devices, the OUT output link field must specify the address of the I/O card, and the DTYP field must specify the name of the corresponding device support module. You can see a list of the device support modules currently supported at the user's local site by using the `dbst` utility in R3.13.

For soft records, the OUT output link can be a constant, a database link, or a channel access link. If the link is a constant, the result is no output. The DTYP field must then specify the `Soft Channel` device support routine.

See *Address Specification, Chapter 1, 2*, for information on the format of hardware addresses and database links.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor
OUT	Output Link	OUTLINK	Yes	0	No	No	N/A
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No

5. Operator Display Parameters

These parameters are used to present meaningful data to the operator. They display the value and other parameters of the long output either textually or graphically.

EGU is a string of up to 16 characters describing the units that the long output measures. It is retrieved by the `get_units` record support routine.

The HOPR and LOPR fields set the upper and lower display limits for the VAL, HIHI, HIGH, LOW, and LOLO fields. Both the `get_graphic_double` and `get_control_double` record support routines retrieve these fields.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	null	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

The possible alarm conditions for long inputs are the SCAN, READ, INVALID, and limit alarms. The SCAN and READ alarms are not configurable by the user because their severity is always MAJOR. The INVALID alarm is called by the record support routine when the record or device support routines cannot write the record's output. The IVOA field specifies the action to take in this case.

The limit alarms are configured by the user in the HIHI, LOLO, HIGH, and LOW fields using floating-point values. For each of these fields, there is a corresponding severity field which can be either NO_ALARM, MINOR, or MAJOR. The HYST field contains the alarm deadband around each limit alarm.

See the See *Alarm Specification, Chapter 1, 4*, for a complete explanation of alarms and these fields. For an explanation of the IVOA and IVOV fields, see *Output Records, Chapter 3, 3*. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HIHI	Hihi Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HIGH	High Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOW	Low Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOLO	Lolo Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HHSV	Hihi Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HSV	High Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LSV	Low Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LLSV	Lolo Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HYST	Alarm Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
IVOA	Invalid Alarm Output Action	GBLCHOICE	Yes	0	Yes	Yes	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
IVOV	Invalid Alarm Output Value, in eng. units	DOUBLE	Yes	0	Yes	Yes	No	No

7. Monitor Parameters

These parameters are used to determine when to send monitors placed on the value field. The monitors are sent when the value field exceeds the last monitored field by the appropriate delta. If these fields have a value of zero, everytime the value changes, a monitor will be triggered; if they have a value of -1, everytime the record is scanned, monitors are triggered. The ADEL field is the delta for archive monitors, and the MDEL field is the delta for all other types of monitors. See *Monitor Specification, Chapter 1, 5*, for a complete explanation of monitors.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ADEL	Archive Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
MDEL	Monitor, i.e. value change, Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

8. Run-time and Simulation Mode Parameters

The LALM, MLST, and ALST fields are used to implement the hysteresis factors for monitor callbacks. Only if the difference between these fields and the corresponding value field is greater than the appropriate delta (MDEL, ADEL, HYST)—only then are monitors triggered. For instance, only if the difference between VAL and MLST is greater than MDEL are the monitors triggered for VAL.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LALM	Last Alarm Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No
ALST	Last Archiver Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No
MLST	Last Value Change Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No

The following fields are used to operate the long output in the simulation mode. See Chapter 3, *Fields Common to Many Record Types*, for more information on the simulation mode fields

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SVAL	Simulation Value	DOUBLE	No	0	Yes	Yes	No	No
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

9. Record Support Routines

init_record

This routine initializes SIMM if SIML is a constant or creates a channel access link if SIML is PV_LINK. If SIOL is PV_LINK a channel access link is created.

This routine next checks to see that device support is available. The routine next checks to see if the device support write routine is defined.

If either device support or the device support write routine does not exist, an error message is issued and processing is terminated.

If DOL is a constant, then VAL is initialized to its value and UDF is set to FALSE. If DOL type is a PV_LINK then dbCaAddInlink is called to create a channel access link.

If device support includes init_record, it is called.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_units

Retrieves EGU.

get_graphic_double

Sets the upper display and lower display limits for a field. If the field is VAL, HIHI, HIGH, LOW, or LOLO, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_control_double

Sets the upper control and the lower control limits for a field. If the field is VAL, HIHI, HIGH, LOW, or LOLO, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_alarm_double

Sets the following values:

upper_alarm_limit = HIHI

upper_warning_limit = HIGH

lower_warning_limit = LOW

lower_alarm_limit = LOLO

10. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. If PACT is FALSE and OMSL is CLOSED_LOOP recGblGetLinkValue is called to read the current value of VAL. See *Output Records, Chapter 3, 3*, for more information. If the return status of recGblGetLinkValue is zero then UDF is set to FALSE.
3. Check alarms. This routine checks to see if the new VAL causes the alarm status and severity to change. If so, NSEV, NSTA and LALM are set. It also honors the alarm hysteresis factor (HYST). Thus the value must change by more than HYST before the alarm status and severity is lowered.
4. Check severity and write the new value. See *Invalid Alarm Output Action, Chapter 3, 3.5*, for information on how INVALID alarms affect output records.
5. If PACT has been changed to TRUE, the device support write output routine has started but has not completed writing the new value. In this case, the processing routine merely returns, leaving PACT TRUE.
6. Check to see if monitors should be invoked.
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - Archive and value change monitors are invoked if ADEL and MDEL conditions are met.
    - NSEV and NSTA are reset to 0.
7. Scan forward link if necessary, set PACT FALSE, and return.

11. Device Support

11.1. Fields Of Interest To Device Support

Each long output record must have an associated set of device support routines. The primary responsibility of the device support routines is to output a new value whenever write_longout is called. The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See Chapter 2, *Fields Common to All Record Types* for an explanation of these fields.
DPVT	Device Private	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
OUT	Output Link	This field is used by the device support routines to locate its output.

11.2. Device Support Routines

Device support consists of the following routines:

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
    *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

write_longout

```
write_longout(precord)
```

This routine must output a new value. It returns the following values:

•0: Success.

•Other: Error.

11.3. Device Support For Soft Records

The `Soft Channel` module writes the current value of VAL.

If the OUT link type is PV_LINK, then dbCaAddInlink is called by init_record.

write_longout calls recGblPutLinkValue to write the current value of VAL.

See *Soft Output, Chapter 3, 3.2*, for a further explanation.

Chapter 19:mbbi — Multi-Bit Binary Input

1. Introduction

The normal use for the multi-bit binary input record is to read multiple bit inputs from hardware. The binary value represents a state from a range of up to 16 states. The multi-bit input record interfaces with devices that use more than one bit.

Most device support modules obtain values from hardware and place the value in RVAL. For these device support modules record processing uses RVAL to determine the current state (VAL is given a value between 0 and 15). Device support modules may optionally read a value directly into VAL.

Soft device modules are provided to obtain input via database or channel access links or via dbPutField or dbPutLink requests. Two soft device support modules are provided: `Soft Channel` allows VAL to be an arbitrary unsigned short integer. `Raw Soft Channel` reads the value into RVAL just like normal device support modules.

The multi-bit binary input fields fall into the following categories:

scan parameters

read and convert parameters

operator display parameters

alarm parameters

run-time and simulation mode parameters

2. Scan parameters

The multi-bit binary input record has the standard fields for specifying under what circumstances it will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Read and Convert Parameters

The device support routines obtain the record's input from the device or link specified in the INP field. For records that obtain their input from devices, the INP field must contain the address of the I/O card, and the DTYP field must specify the proper device support module. Be aware that the address format differs according to the I/O bus used. See *Address Specification, Chapter 1, 2*, for information on the format of hardware addresses. You can see a list of the device support modules currently supported at the user's local site by using the `dbst` utility in R3.13.

Two soft device support modules can be specified in DTYP—`Soft Channel` and `Raw Soft Channel`. `Raw Soft Channel` reads the value into RVAL, upon which the normal conversion process is undergone. `Soft Channel` reads any unsigned integer directly into VAL. For a soft mbbi record, the INP field can be a constant, a database, or a channel access link. If INP is a constant, then the VAL is initialized to the constant value but can be changed at run-time via dbPutField or dbPutLink. See *Address Specification, Chapter 1, 2*, for information on the format of database addresses.

Unless the device support routine specifies no conversion, RVAL is used to determine VAL as follows:

1. RVAL is assigned to a temporary variable—rval = RVAL
2. rval is shifted right SHFT number of bits.
3. A match is sought between rval and one of the state value fields, ZRVL-FFVL.

Each of the fields, ZRVL-FFVL, represents one of the possible sixteen states (not all sixteen have to be used).

Alternatively, the input value can be read as a string, in which case, a match is sought with one of the strings specified in the ZRST-FFST fields. Then RVAL is set equal to the corresponding value for that string, and the conversion process occurs.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Value Field	ENUM	No	0	Yes	Yes	Yes	Yes
INP	Input Link	INLINK	Yes	0	No	No	N/A	No
RVAL	Raw Data Value	ULONG	No	0	Yes	Yes	Yes	Yes
SHFT	Shift	USHORT	No	0	Yes	No	No	No
ZRVL	Zero Value	ULONG	Yes	0	Yes	Yes	No	Yes
ONVL	One value	ULONG	Yes	0	Yes	Yes	No	Yes
TWVL	Two Value	ULONG	Yes	0	Yes	Yes	No	Yes
THVL	Three Value	ULONG	Yes	0	Yes	Yes	No	Yes
FRVL	Four Value	ULONG	Yes	0	Yes	Yes	No	Yes
FVVL	Five Value	ULONG	Yes	0	Yes	Yes	No	Yes
SXVL	Six Value	ULONG	Yes	0	Yes	Yes	No	Yes
SVVL	Seven Value	ULONG	Yes	0	Yes	Yes	No	Yes
EIVL	Eight value	ULONG	Yes	0	Yes	Yes	No	Yes
NIVL	Nine Value	ULONG	Yes	0	Yes	Yes	No	Yes
TEVL	Ten Value	ULONG	Yes	0	Yes	Yes	No	Yes

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ELVL	Eleven Value	ULONG	Yes	0	Yes	Yes	No	Yes
TVVL	Twelve Value	ULONG	Yes	0	Yes	Yes	No	Yes
TTVL	Thirteen Value	ULONG	Yes	0	Yes	Yes	No	Yes
FTVL	Fourteen Value	ULONG	Yes	0	Yes	Yes	No	Yes
FFVL	Fifteen Value	ULONG	Yes	0	Yes	Yes	No	Yes
ZRST	Zero String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
ONST	One String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
TWST	Two String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
THST	Three String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
FRST	Four String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
FVST	Five String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
SXST	Six String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
SVST	Seven String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
EIST	Eight String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
NIST	Nine String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
TEST	Ten String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
ELST	Eleven String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
TVST	Twelve String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
TTST	Thirteen String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
FTST	Fourteen String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
FFST	Fifteen String	STRING [16]	Yes	Null	Yes	Yes	No	Yes

4. Operator Display Parameters

These parameters are used to present meaningful data to the operator. They display the value and other parameters of the mbbi record either textually or graphically. The ZRST-FFST fields contain strings describing one of the possible states of the record. The `get_enum_str` and `get_enum_strs` record routines retrieve these strings for the operator. `Get_enum_str` gets the string corresponding to the value set in VAL, and `get_enum_strs` retrieves all the strings.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ZRST,..., FFST	Zero String, One String, ...	STRING [16]	Yes	Null	Yes	Yes	No	Yes
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

5. Alarm Parameters

The possible alarm conditions for multi-bit binary inputs are the SCAN, READ, and state alarms. The state alarms are configured in the below severity fields. These fields have the usual possible values for severity fields: NO ALARM, MINOR, and MAJOR.

The unknown state severity (UNSV) field, if set to MINOR or MAJOR, triggers an alarm when the record support routine cannot find a matching value in the state value fields for `rval`.

The change of state severity (COSV) field triggers an alarm when any change of state occurs, if set to MAJOR or MINOR.

The other fields, when set to MAJOR or MINOR, trigger an alarm when VAL equals the corresponding state. See the See *Alarm Specification, Chapter 1, 4*, for a complete explanation of discrete alarms and these fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
UNSV	Unknown State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
COSV	Change of State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
ZRSV	0 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
ONSV	1 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
TWSV	2 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
THSV	3 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
FRSV	4 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
FVSV	5 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
SXSV	6 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
SVSV	7 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
EISV	8 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
NISV	9 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
TESV	10 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
ELSV	11 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
TVSV	12 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
TTSV	13 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
FTSV	14 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
FFSV	15 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes

6. Run-time and Simulation Mode Parameters

These parameters are used by the run-time code for processing the multi-bit binary input.

MASK is used by device support routine to read the hardware register. It is shifted left NOBT bits by the record processing routine. The user can configure the NOBT field, but the device support routines may set it, in which case the value given to it by the user is simply overridden.

The LALM field implements the change of state alarm severity by holding the value of VAL when the previous change of state alarm was issued.

MLST holds the value when the last monitor for value change was triggered.

SDEF is used by record support to save time if no states are defined.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SHFT	Shift	USHORT	No	0	Yes	No	No	No
ORAW	Old Raw Data	ULONG	No	0	Yes	No	No	No
NOBT	Number of Bits	SHORT	Yes	0	Yes	No	No	No
MASK	Mask	ULONG	No	0	Yes	No	No	No
LALM	Last Alarmed	USHORT	No	0	Yes	No	No	No
MLST	Monitor Last	USHORT	No	0	Yes	No	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SDEF	States Defined?	SHORT	No	0	Yes	No	No	No

The following fields are used to operate the mbbi record in the simulation mode. See Chapter 3, *Fields Common to Many Record Types*, for more information on these fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No
SVAL	Simulation Value	DOUBLE	No	0	Yes	Yes	No	No
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

7. Record Support Routines

init_record

This routine initializes SIMM with the value of SIML if SIML type is CONSTANT link or creates a channel access link if SIML type is PV_LINK. SVAL is likewise initialized if SIOL is CONSTANT or PV_LINK.

This routine next checks to see that device support is available and a device support read routine is defined. If either does not exist, an error message is issued and processing is terminated.

Clears MASK and then sets the NOBT low order bits.

If device support includes init_record, it is called.

init_common is then called to determine if any states are defined. If states are defined, SDEF is set to TRUE.

process

See next section.

special

Calls init_common to compute SDEF when any of the fields ZRVL, ... FFVL change value.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_enum_str

Retrieves ASCII string corresponding to VAL.

get_enum_strs

Retrieves ASCII strings for ZRST,...FFST.

put_enum_str

Checks if string matches ZRST,...FFST and if it does, sets VAL.

8. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. readValue is called. See *Input Records, Chapter 3, 2* for more information.
3. If PACT has been changed to TRUE, the device support read routine has started but has not completed reading a new input value. In this case, the processing routine merely returns, leaving PACT TRUE.
4. Convert.
    - status=read_mbbi
    - PACT = TRUE
    - TIME = tsLocalTime
    - If status is 0, then determine VAL
        - Set rval = RVAL
        - Shift rval right SHFT bits
    - If at least one state value is defined
        - Set UDF to TRUE
    - If RVAL is ZRVL,...,FFVL then set
        - VAL equals index of state
        - UDF set to FALSE
    - Else set VAL = undefined
        - Else set VAL = RVAL
    - Set UDF to FALSE
        - If status is 1, return(0)
    - If status is 2, set status = 0
5. Check alarms. This routine checks to see if the new VAL causes the alarm status and severity to change. If so, NSEV, NSTA and LALM are set.
6. Check to see if monitors should be invoked.

- Alarm monitors are invoked if the alarm status or severity has changed.
- Archive and value change monitors are invoked if MLST is not equal to VAL.
- Monitors for RVAL are checked whenever other monitors are invoked.
- NSEV and NSTA are reset to 0.

7. Scan forward link if necessary, set PACT FALSE, and return.

9. Device Support

9.1. Fields Of Interest To Device Support

Each input record must have an associated set of device support routines.

The primary responsibility of the device support routines is to obtain a new raw input value whenever read_mbbi is called. The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See Chapter 2, *Fields Common to All Record Types* for an explanation of these fields.
DPVT	Device Private	
UDF	VAL Undefined	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
NOBT	Number of Bits	Number of hardware bits accessed. They must be consecutive.
VAL	Value Field	This field is set by the device support routines if they don't want record support to set it.
INP	Input Link	This field is used by the device support routines to locate its input.
RVAL	Raw Data Value	It is the responsibility of the device support routine to give this field a value.
MASK	Mask	This is a mask used to read the hardware. Record support sets the low order NOBT bits. The device support routine can shift the bits. The device support routine should perform the shift in init_record.
SHFT	Shift	This can be set by the device support module at init_record time.

9.2. Device Support Routines

Device support consists of the following routines:

report

```
report(FILE fp, paddr)
```

Not currently used.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine. If it uses MASK, it should shift it as necessary and also give SHFT a value.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
     *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the I/O Event scanner.

read_mbbi

```
read_mbbi(precord)
```

This routine must provide a new input value. It returns the following values:

- 0: Success. A new raw value is placed in RVAL. The record support module determines VAL from RVAL, SHFT, and ZEVL ... FFVL.
- 2: Success, but don't modify VAL.
- Other: Error.

9.3. Device Support For Soft Records

Two soft device support modules `Soft Channel` and `Raw Soft Channel` are provided for multi-bit binary input records not related to actual hardware devices. The INP link type must be either CONSTANT, DB_LINK, or CA_LINK.

Soft Channel

read_mbbi always returns a value of 2, which means that no conversion is performed.

If the INP link type is constant, then the constant value is stored into VAL by init_record, and UDF is set to FALSE. VAL can be changed via dbPut requests. If the INP link type is PV_LINK, then dbCaAddInlink is called by init_record.

read_mbbi calls recGblGetLinkValue to read the current value of VAL. See *Soft Input, Chapter 3, 2.3*.

If the return status of recGblGetLinkValue is zero, then read_mbbi sets UDF to FALSE. The status of recGblGetLinkValue is returned.

Raw Soft Channel

This module is like the previous except that values are read into RVAL, VAL is computed from RVAL, and read_mbbi returns a value of 0. Thus the record processing routine will determine VAL in the normal way.

Chapter 20:mbbiDirect - *Multi-Bit Binary Input Direct*

Johnny Tang, Matthew Bickley, and Chip Watson
Continuous Electron Beam Accelerator Facility
Southeastern Universities Research Association

1. Introduction

The mbbiDirect record retrieves a sixteen-bit hardware value and converts it to an array of sixteen unsigned characters, each representing a bit of the word. These fields (B0-B9, BA-BF) are set to 1 if a bit is set, and 0 if not.

This record's operation is similar to that of the multi-bit binary input record, and it has many fields in common with it. This record also has two available soft device support modules—`Soft Channel` and `Raw Soft Channel`. The fields fall into the following categories:

scan parameters

read and convert parameters

operator display parameters

alarm parameters

run-time and simulation mode parameters

2. Scan fields

The mbbiDirect record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Read and Convert fields

The device support routines obtain the record's input from the device or link specified in the INP field. For records that obtain their input from devices, the INP field must contain the address of the I/O card, and the DTYP field must specify the proper device support module. Be aware that the address format differs according to the I/O bus used. See *Address Specification, Chapter 1, 2*, for information on the format of hardware addresses. You can see a list of the device support modules currently supported at the user's local site by using the `dbst` utility in R3.13.

Two soft device support modules can be specified in DTYP—`Soft Channel` and `Raw Soft Channel`. `Raw Soft Channel` reads the value into RVAL, upon which the normal conversion process is undergone. `Soft Channel` reads any unsigned integer directly into VAL. For a soft mbbiDirect record, the INP field can be a constant, a database, or a channel access link. If INP is a constant, then the VAL is initialized to the INP value but can be changed at run-time via dbPutField or dbPutLink. See *Address Specification, Chapter 1, 2*, for information on how to database links.

For records that don't use `Soft Channel` device support, RVAL is used to determine VAL as follows:

1. RVAL is assigned to a temporary variable—rval = RVAL
2. rval is shifted right SHFT number of bits.
3. VAL is set equal to rval.

Each of the fields, B0-BF, represents one bit of the word.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Value Field	ENUM	No	0	Yes	Yes	Yes	Yes
INP	Input Link	INLINK	Yes	0	No	No	N/A	No
RVAL	Raw Data Value	ULONG	No	0	Yes	Yes	Yes	Yes
SHFT	Shift	USHORT	No	0	Yes	No	No	No
B0	Bit 0 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B1	Bit 1 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B2	Bit 2 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B3	Bit 3 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B4	Bit 4 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B5	Bit 5 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B6	Bit 6 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B7	Bit 7 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B8	Bit 8 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B9	Bit 9 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
BA	Bit 10 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
BB	Bit 12 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
BC	Bit 13 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
BD	Bit 14 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
BE	Bit 15 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
BF	Bit 16 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes

4. Operator Display Parameters

These parameters are used to present meaningful data to the operator.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

5. Alarm Parameters

The possible alarm conditions for multi-bit binary input direct records are the SCAN and READ alarms. These alarms are not configurable by the user since they are always of MAJOR severity. See *Alarm Specification, Chapter 1, 4*, for a complete explanation of Scan and Read alarms. No fields exist for the mbbi direct record to have state alarms.

Alarm Fields, Chapter 2, 3, lists other fields related to a alarms that are common to all record types.

6. Run-time and Simulation Mode Fields

These parameters are used by the run-time code for processing the mbbi direct record. They are not configurable prior to run-time.

MASK is used by device support routine to read hardware register. Record support sets low order NOBT bits in MASK. Device support can shift this value.

The LALM field implements the change of state alarm severity by holding the value of VAL when the previous change of state alarm was issued.

MLST holds the value when the last monitor for value change was triggered.

SDEF is used by record support to save time if no states are defined.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NOBT	Number of Bits	SHORT	Yes	0	Yes	No	No	No
ORAW	Old Raw Data	ULONG	No	0	Yes	No	No	No
MASK	Mask	ULONG	No	0	Yes	No	No	No
LALM	Last Alarmed	USHORT	No	0	Yes	No	No	No
MLST	Monitor Last	USHORT	No	0	Yes	No	No	No
SDEF	States Defined?	SHORT	No	0	Yes	No	No	No

The following fields are used to operate the mbbiDirect record in the simulation mode. See Chapter 3, *Fields Common to Many Record Types*, for more information on these fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No
SVAL	Simulation Value	DOUBLE	No	0	Yes	Yes	No	No
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

7. Record Support Routines

init_record

This routine initializes SIMM with the value of SIML if SIML type is CONSTANT link or creates a channel access link if SIML type is PV_LINK. SVAL is likewise initialized if SIOL is CONSTANT or PV_LINK.

This routine next checks to see that device support is available and a device support read routine is defined. If either does not exist, an error message is issued and processing is terminated.

Clears MASK and then sets the NOBT low order bits.

If device support includes init_record, it is called.

refresh_bits is then called to refresh all the bit fields based on a hardware value.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

8. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. readValue is called. See *Output Records, Chapter 3, 3*, for information.
3. If PACT has been changed to TRUE, the device support read routine has started but has not completed reading a new input value. In this case, the processing routine merely returns, leaving PACT TRUE.
4. Convert.
   - status=read_mbbiDirect
   - PACT = TRUE
   - TIME = tsLocalTime
   - If status is 0, then determine VAL
     - Set rval = RVAL
     - Shift rval right SHFT bits
     - Set VAL = RVAL
     - If status is 1, return(0)
   - If status is 2, set status = 0
5. Check to see if monitors should be invoked.
   - Alarm monitors are invoked if the alarm status or severity has changed.
   - Archive and value change monitors are invoked if MLST is not equal to VAL.
   - Monitors for RVAL are checked whenever other monitors are invoked.
   - NSEV and NSTA are reset to 0.
6. Scan forward link if necessary, set PACT FALSE, and return.

9. Device Support

9.1. Fields Of Interest To Device Support

Each input record must have an associated set of device support routines.

The primary responsibility of the device support routines is to obtain a new raw input value whenever read_mbbiDirect is called. The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See Chapter 2, *Fields Common to All Record Types* for an explanation of these fields.
DPVT	Device Private	
UDF	VAL Undefined	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
NOBT	Number of Bits	Number of hardware bits accessed. They must be consecutive.

Name	Summary	Description
VAL	Value Field	This field is set by the device support routines if they don't want record support to set it.
INP	Input Link	This field is used by the device support routines to locate its input.
RVAL	Raw Data Value	It is the responsibility of the device support routine to give this field a value.
MASK	Mask	This is a mask used to read the hardware. Record support sets the low order NOBT bits. The device support routine can shift the bits. The device support routine should perform the shift in init_record.
SHFT	Shift	This can be set by the device support module at init_record time.

9.2. Device Support Routines

Device support consists of the following routines:

report

```
report(FILE fp, paddr)
```

Not currently used.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine. If it uses MASK, it should shift it as necessary and also give SHFT a value.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
    *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

read_mbbiDirect

```
read_mbbiDirect(precord)
```

This routine must provide a new input value. It returns the following values:

- 0: Success. A new raw value is placed in RVAL. The record support module determines VAL from RVAL and SHFT.
- 2: Success, but don't modify VAL.
- Other: Error.

9.3. Device Support For Soft Records

Two soft device support modules, `Soft Channel` and `Raw Soft Channel`, are provided for multi-bit binary input direct records not related to actual hardware devices. The INP link type must be either CONSTANT, DB_LINK, or CA_LINK.

Soft Channel

For this module, read_mbbiDirect always returns a value of 2, which means that no conversion is performed.

If the INP link type is constant, then the constant value is stored into VAL by init_record, and UDF is set to FALSE. VAL can be changed via dbPut requests. If the INP link type is PV_LINK, then dbCaAddInlink is called by init_record.

read_mbbiDirect calls recGblGetLinkValue to read the current value of VAL.

See *Input Records, Chapter 3, 2*, for a further explanation.

If the return status of recGblGetLinkValue is zero, then read_mbbi sets UDF to FALSE. The status of recGblGetLinkValue is returned.

Raw Soft Channel

This module is like the previous except that values are read into RVAL, VAL is computed from RVAL, and read_mbbiDirect returns a value of 0. Thus the record processing routine will determine VAL in the normal way.

Chapter 21:mbbo — Multi-Bit Binary Output

1. Introduction

The normal use for the mbbo record type is to send a binary value (representing one of up to 16 states) to a Digital Output module. It is used for any device that uses more than one bit to control it. The mbbo record can also be used to write discrete values to other records via database or channel access links.

The multi-bit binary output fields fall into the following categories:

- •scan parameters
- •desired output parameters
- •write and convert parameters
- •operator display parameters
- •alarm parameters
- •run-time parameters

2. Scan Parameters

The mbbo record has the standard fields for specifying under what circumstances it will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Desired Output Parameters

The multi-bit binary output record, like all output records, must specify where its output originates. The output mode select (OMSL) field determines whether the output originates from another record or from database access (i.e., the operator). When set to `closed_loop`, the desired output is retrieved from the link specified in the desired output (DOL) field—which can specify either a database or channel access link—and placed into the VAL field. When set to `supervisory`, the DOL field is ignored and the current value of VAL is simply written. VAL can be changed via dpPuts at run-time when OMSL is `supervisory`. The DOL field can also be a constant, in which case the VAL field is initialized to the constant value. If DOL is a constant, OMSL cannot be set to `closed_loop`.

The VAL field itself usually consists of an index that specifies one of the states. The actual output written is the value of RVAL, which is converted from VAL following the routine explained in the next section. However, records that use the `Soft Channel` device support module write the VAL field's value without any conversion.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OMSL	Output Mode Select	GBLCHOICE	Yes	0	Yes	Yes	No	No
DOL	Desired Output Location (an Input Link)	INLINK	Yes	0	No	No	N/A	No
VAL	Value Field	ENUM	No	0	Yes	Yes	Yes	Yes

4. Convert and Write Parameters

The device support routines write the desired output to the location specified in the OUT field. If the record uses soft device support, OUT can contain a constant, a database link, or a channel access link; however, if OUT is a constant, no value will be written.

For records that write their values to hardware devices, the OUT output link must specify the address of the I/O card, and the DTYP field must specify the corresponding device support module. Be aware that the address format differs according to the I/O bus used. See *Address Specification, Chapter 1, 2*, for information on the format of hardware addresses. The user can see a list of the device support modules currently supported at the user's local site by using the `dbst` utility in R3.13.

For mbbo records that write to hardware, the value written to the output location is the value contained in RVAL, which is converted from VAL, VAL containing an index of one of the 16 states (0-15). RVAL is then set to the corresponding state value, the value in one of the fields ZRVL through FFVL. Then this value is shifted left according to the number in the SHFT field so that the value is in the correct position for the bits being used (the SHFT value is set by device support and is not configurable by the user).

The state value fields ZRVL through FFVL must be configured by the user before run-time. When the state values are not defined, the states defined (SDEF) field is set to FALSE at initialization time by the record routines. When SDEF is FALSE, then the record processing routine does not try to find a match, RVAL is set equal to VAL, the bits are shifted using the number in SHFT, and the value is written thus.

If the OUT output link specifies a database link, channel access link, or constant, then the DTYP field must specify either one of the two soft device support modules—`Soft Channel` or `Raw Soft Channel`. `Soft Channel` writes the value of VAL to the output link, without any conversion, while `Raw Soft Channel` writes the value from RVAL after it has undergone the above conversion. See *Address Specification, Chapter 1, 2*, for information on specifying links.

Note also that when a string is retrieved as the desired output, a record support routine is provided (`put_enum_str`) that will set check to see if the string matches one of the strings in the ZRST through FFST fields. If a match is found, RVAL is set equal to the corresponding state value of that string.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OUT	Output Link	OUTLINK	Yes	0	No	No	N/A	No
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
RVAL	Raw Data Value	ULONG	No	0	Yes	Yes	Yes	Yes
SHFT	Shift	USHORT	No	0	Yes	No	No	No
SDEF	States Defined?	SHORT	No	0	Yes	No	No	No
ZRVL	Zero Value	ULONG	Yes	0	Yes	Yes	No	Yes
ONVL	One value	ULONG	Yes	0	Yes	Yes	No	Yes
TWVL	Two Value	ULONG	Yes	0	Yes	Yes	No	Yes
THVL	Three Value	ULONG	Yes	0	Yes	Yes	No	Yes
FRVL	Four Value	ULONG	Yes	0	Yes	Yes	No	Yes
FVVL	Five Value	ULONG	Yes	0	Yes	Yes	No	Yes
SXVL	Six Value	ULONG	Yes	0	Yes	Yes	No	Yes
SVVL	Seven Value	ULONG	Yes	0	Yes	Yes	No	Yes
EIVL	Eight value	ULONG	Yes	0	Yes	Yes	No	Yes
NIVL	Nine Value	ULONG	Yes	0	Yes	Yes	No	Yes
TEVL	Ten Value	ULONG	Yes	0	Yes	Yes	No	Yes
ELVL	Eleven Value	ULONG	Yes	0	Yes	Yes	No	Yes
TVVL	Twelve Value	ULONG	Yes	0	Yes	Yes	No	Yes
TTVL	Thirteen Value	ULONG	Yes	0	Yes	Yes	No	Yes
FTVL	Fourteen Value	ULONG	Yes	0	Yes	Yes	No	Yes
FFVL	Fifteen Value	ULONG	Yes	0	Yes	Yes	No	Yes
ZRST	Zero String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
ONST	One String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
TWST	Two String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
THST	Three String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
FRST	Four String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
FVST	Five String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
SXST	Six String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
SVST	Seven String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
EIST	Eight String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
NIST	Nine String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
TEST	Ten String	STRING [16]	Yes	Null	Yes	Yes	No	Yes

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ELST	Eleven String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
TVST	Twelve String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
TTST	Thirteen String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
FTST	Fourteen String	STRING [16]	Yes	Null	Yes	Yes	No	Yes
FFST	Fifteen String	STRING [16]	Yes	Null	Yes	Yes	No	Yes

5. Operator Display Parameters

These parameters are used to present meaningful data to the operator. These fields are used to display the value and other parameters of the mbbo record either textually or graphically. The ZRST-FFST fields contain strings describing each of the corresponding states. The `get_enum_str` and `get_enum_strs` record routines retrieve these strings for the operator. `Get_enum_str` gets the string corresponding to the value set in VAL, and `get_enum_strs` retrieves all the strings.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ZRST,..., FFST	Zero String, One String, ...	STRING [16]	Yes	Null	Yes	Yes	No	Yes
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

The possible alarm conditions for multi-bit binary outputs are the SCAN, READ, INVALID, and state alarms. The SCAN and READ alarms are called by the support modules and are not configurable by the user, as their severity is always MAJOR.

The IVOA field specifies an action to take from a number of possible choices when the INVALID alarm is triggered. The IVOV field contains a value to be written once the INVALID alarm has been triggered if `Set output to IVOV` has been chosen in the IVOA field. The severity of the INVALID alarm is not configurable by the user.

The state alarms are configured in the below severity fields. These fields have the usual possible values for severity fields: NO ALARM, MINOR, and MAJOR.

The unknown state severity field (UNSV), if set to MINOR or MAJOR, triggers an alarm when the record support routine cannot find a matching value in the state value fields for VAL or when VAL is out of range.

The change of state severity field (COSV) triggers an alarm when the record's state changes, if set to MAJOR or MINOR.

The state severity (ZRSV-FFSV) fields, when set to MAJOR or MINOR, trigger an alarm when VAL equals the corresponding field.

See *Alarm Specification, Chapter 1, 4*, for a complete explanation of discrete alarms and these fields. See *Invalid Alarm Output Action, Chapter 3, 3.5*, for an explanation of the IVOA and IVOV fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
UNSV	Unknown State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
COSV	Change of State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
IVOA	Invalid Alarm Output Action	GBLCHOICE	Yes	0	Yes	Yes	No	No
IVOV	Invalid Alarm Output Value, in eng. units	DOUBLE	Yes	0	Yes	Yes	No	No
ZRSV	0 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
ONSV	1 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
TWSV	2 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
THSV	3 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
FRSV	4 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
FVSV	5 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
SXSV	6 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
SVSV	7 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
EISV	8 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
NISV	9 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
TESV	10 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
ELSV	11 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
TVSV	12 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
TTSV	13 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
FTSV	14 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
FFSV	15 State Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes

7. Run-Time and Simulation Mode Parameters

These parameters are used by the run-time code for processing the multi-bit binary output.

MASK is used by device support routine to read the hardware register. Record support sets low order of MASK the number of bits specified in NOBT. Device support can shift this value.

The LALM field implements the change of state alarm severity by holding the value of VAL when the previous change of state alarm was issued.

MLST holds the value when the last monitor for value change was triggered.

SDEF is used by record support to save time if no states are defined; it is used for converting VAL to RVAL.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NOBT	Number of Bits	SHORT	Yes	0	Yes	No	No	
ORAW	Old Raw Data	ULONG	No	0	Yes	No	No	No
MASK	Mask	ULONG	No	0	Yes	No	No	No
LALM	Last Alarmed	USHORT	No	0	Yes	No	No	No
MLST	Monitor Last	USHORT	No	0	Yes	No	No	No
SDEF	States Defined?	SHORT	No	0	Yes	No	No	No

The following fields are used to operate the mbbo record in the simulation mode. See Chapter 3, *Fields Common to Many Record Types*, for more information on the simulation mode fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No
SVAL	Simulation Value	DOUBLE	No	0	Yes	Yes	No	No
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

8. Record Support Routines

init_record

This routine initializes SIMM if SIML is a constant or creates a channel access link if SIML is PV_LINK. If SIOL is PV_LINK a channel access link is created.

This routine next checks to see that device support is available. The routine next checks to see if the device support write routine is defined. If either device support or the device support write routine does not exist, an error message is issued and processing is terminated.

If DOL is a constant, then VAL is initialized to its value and UDF is set to FALSE.

MASK is cleared and then the NOBT low order bits are set.

If device support includes init_record, it is called.

init_common is then called to determine if any states are defined. If states are defined, SDEF is set to TRUE.

If device support returns success, VAL is then set from RVAL and UDF is set to FALSE.

process

See next section.

special

Computes SDEF when any of the fields ZRVL,...FFVL change value.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_enum_str

Retrieves ASCII string corresponding to VAL.

get_enum_strs

Retrieves ASCII strings for ZRST,...FFST.

put_enum_str

Checks if string matches ZRST,...FFST and if it does, sets VAL.

9. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will not longer be called for this record. Thus error storms will not occur.
2. If PACT is FALSE
   - If DOL is DB_LINK and OMSL is CLOSED_LOOP
     - Get value from DOL
     - Set UDF to FALSE
     - Check for link alarm
   - If any state values are defined
     - If VAL > 15, then raise alarm and go to 4

        - Else using VAL as index set RVAL = one of ZRVL,...FFVL
    - Else set RVAL = VAL
    - Shift RVAL left SHFT bits
3. Convert
    - If PACT is FALSE, compute RVAL
        - If VAL is 0,...,15, set RVAL from ZRVL,...,FFVL
        - If VAL out of range, set RVAL = undefined
    - Status = write_mbbo
4. Check alarms. This routine checks to see if the new VAL causes the alarm status and severity to change. If so, NSEV, NSTA and LALM are set.
5. Check severity and write the new value. See *Simulation Mode, Chapter 3, 3.4*, and *Invalid Alarm Output Action, Chapter 3, 3.5*, for more information.
6. If PACT has been changed to TRUE, the device support write output routine has started but has not completed writing the new value. In this case, the processing routine merely returns, leaving PACT TRUE.
7. Check to see if monitors should be invoked.
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - Archive and value change monitors are invoked if MLST is not equal to VAL.
    - Monitors for RVAL and RBV are checked whenever other monitors are invoked.
    - NSEV and NSTA are reset to 0.
8. Scan forward link if necessary, set PACT FALSE, and return.

10. Device Support

10.1. Fields Of Interest To Device Support

Each mbbo record must have an associated set of device support routines. The primary responsibility of the device support routines is to obtain a new raw mbbo value whenever write_mbbo is called. The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See Chapter 2, *Fields Common to All Record Types* for an explanation of these fields.
DPVT	Device Private	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
NOBT	Number of Bits	Number of hardware bits accessed. They must be consecutive.
OUT	Output Link	This field is used by the device support routines to locate its output.

Name	Summary	Description
RVAL	Raw data value.	This is the value to be written to OUT.
RBV	Read Back Value	It is the responsibility of the device support modules to set this field.
MASK	Mask	This is a mask used to read the hardware. Record support sets the low order NOBT bits. The device support routine can shift the bits. The device support routine should perform the shift in init_record.
SHFT	Shift	This can be set by the device support module at init_record time.

10.2. Device Support Routines

Device support consists of the following routines:

report

```
report(FILE fp, paddr)
```

Not currently used.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine. If MASK is used, it should be shifted if necessary and SHFT given a value.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
     *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

write_mbbo

```
write_mbbo(precord)
```

This routine must output a new value. It returns the following values:

- •0: Success.
- •Other: Error.

10.3. Device Support For Soft Records

Soft Channel

The `Soft Channel` module writes the current value of VAL.

If the OUT link type is PV_LINK, then dbCaAddInlink is called by init_record.

write_mbbo calls recGblPutLinkValue to write the current value of VAL. See *Soft Output, Chapter 3, 3.2*, for more information.

Raw Soft Channel

This module writes RVAL to the location specified in the output link. It returns a 0.

Chapter 22:mbboDirect - Multi-Bit Binary Output Direct

Johnny Tang, Matthew Bickley, and Chip Watson
Continuous Electron Beam Accelerator Facility
Southeastern Universities Research Association

1. Introduction

The mbboDirect record performs the opposite function to that of the mbbiDirect record. It accumulates bits (in the fields B0 - BF) as unsigned characters, and converts them to a word which is then written out to hardware. If a bit field is non-zero, it is interpreted as a binary 1. On the other hand, if it is zero, it is interpreted as a binary 0.

The mbboDirect record's fields fall into the following categories:

- scan parameters
- desired output parameters
- write and convert parameters
- operator display parameters
- alarm parameters
- run-time parameters

2. Scan Parameters

The mbboDirect record has the standard fields for specifying under what circumstances it will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Desired Output Parameters

The mbboDirect record, like all output records, must specify where its output originates. The output mode select field (OMSL) determines whether the output originates from another record or from database access. When set to `closed_loop`, the desired output is retrieved from the link specified in the desired output (DOL) field—which can specify either a database or channel access link—and placed into the VAL field. When set to `supervisory`, the DOL field is ignored and the current value of VAL is used. The desired output can be written into the VAL field via dpPuts at run-time when the record is in `supervisory` mode. DOL can also be a constant, in which case VAL is initialized to the constant value. Note that OMSL cannot be `closed_loop` when DOL is a constant. See *Address Specification, Chapter 1, 2*, for information on how to specify database links.

VAL is then converted to RVAL in the routine described in the next section. However, the `Soft Channel` device support module for the mbboDirect record writes the VAL field's value without any conversion.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OMSL	Output Mode Select	GBLCHOICE	Yes	0	Yes	Yes	No	No
DOL	Desired Output Location (an Input Link)	INLINK	Yes	0	No	No	N/A	No
VAL	Value Field	ENUM	No	0	Yes	Yes	Yes	Yes

4. Convert and Write Parameters

For records that are to write values to hardware devices, the OUT output link must contain the address of the I/O card, and the DTYP field must specify the proper device support module. Be aware that the address format differs according to the I/O bus used. See *Address Specification, Chapter 1, 2*, for information on the format of hardware addresses. You can see a list of the device support modules currently supported at the user's local site by using the dbst utility in R3.13.

If the mbboDirect record does not use the `Soft Channel` device support module, then VAL is converted to RVAL, and RVAL is the actual 16-bit word sent out. RVAL is set equal to VAL and then shifted left by the number of bits specified in the SHFT field (the SHFT value is set by device support and is not configurable by the user). RVAL is then sent out to the location specified in the OUT field.

For mbboDirect records that specify a database link, a channel access link, or a constant, the DTYP field must specify either one of two soft device support routines—`Soft Channel` or `Raw Soft Channel`. The difference between the two is that `Soft Channel` writes the desired output value from VAL directly to the output link while `Raw Soft Channel` writes the value from RVAL to the output link after it has undergone the conversion described above. See *Address Specification, Chapter 1, 2*, for information on how to specify database links.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OUT	Output Link	OUTLINK	Yes	0	No	No	N/A	No
RVAL	Raw Data Value	ULONG	No	0	Yes	Yes	Yes	Yes
SHFT	Shift	USHORT	No	0	Yes	No	No	No
B0	Bit 0 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B1	Bit 1 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B2	Bit 2 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B3	Bit 3 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B4	Bit 4 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B5	Bit 5 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B6	Bit 6 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
B7	Bit 7 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B8	Bit 8 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
B9	Bit 9 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
BA	Bit 10 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
BB	Bit 12 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
BC	Bit 13 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
BD	Bit 14 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
BE	Bit 15 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes
BF	Bit 16 Value	UCHAR	Yes	0	Yes	Yes	Yes	Yes

5. Operator Display Parameters

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

The possible alarm conditions for mbboDirect records are the SCAN, READ, and INVALID alarms. The SCAN and READ alarms are not configurable by the user since they are always of MAJOR severity. See *Alarm Specification, Chapter 1, 4*, for a complete explanation of Scan and Read alarms.

The IVOA field specifies an action to take when the INVALID alarm is triggered. There are three possible actions: `Continue normally`, `Don't drive outputs`, or `Set output to IVOV`. When `Set output to IVOV` is specified and a INVALID alarm is triggered, the record will write the value in the IVOV field to output. See *Invalid Alarm Output Action, Chapter 3, 3.5*, for more information. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

No fields exist for this record to have state alarms.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
IVOA	Invalid Alarm Output Action	GBLCHOICE	Yes	0	Yes	Yes	No	No
IVOV	Invalid Alarm Output Value, in eng. units	DOUBLE	Yes	0	Yes	Yes	No	No

7. Run-time and Simulation Mode Parameters

These parameters are used by the run-time code for processing the mbbo Direct record.

MASK is used by device support routine to read the hardware register. Record support sets low order NOBT bits. Device support can shift this value.

The LALM field implements the change of state alarm severity by holding the value of VAL when the previous change of state alarm was issued.

MLST holds the value when the last monitor for value change was triggered.

SDEF is used by record support to save time if no states are defined.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NOBT	Number of Bits	SHORT	Yes	0	Yes	No	No	No
ORAW	Old Raw Data	ULONG	No	0	Yes	No	No	No
MASK	Mask	ULONG	No	0	Yes	No	No	No
LALM	Last Alarmed	USHORT	No	0	Yes	No	No	No
MLST	Monitor Last	USHORT	No	0	Yes	No	No	No
SDEF	States Defined?	SHORT	No	0	Yes	No	No	No

The following fields are used to operate the mbboDirect record in the simulation mode. See *Simulation Mode, Chapter 3, 3.4*, for more information on the simulation mode fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No
SVAL	Simulation Value	DOUBLE	No	0	Yes	Yes	No	No
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

8. Record Support Routines

init_record

This routine initializes SIMM if SIML is a constant or creates a channel access link if SIML is PV_LINK. If SIOL is PV_LINK a channel access link is created.

This routine next checks to see that device support is available.The routine next checks to see if the device support write routine is defined. If either device support or the device support write routine does not exist, an error message is issued and processing is terminated.

If DOL is a constant, then VAL is initialized to its value and UDF is set to FALSE.

MASK is cleared and then the NOBT low order bits are set.

If device support includes init_record, it is called.

init_common is then called to determine if any states are defined. If states are defined, SDEF is set to TRUE.

If device support returns success, VAL is then set from RVAL and UDF is set to FALSE.

Process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

9. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. If PACT is FALSE
    - If DOL is DB_LINK and OMSL is CLOSED_LOOP
        - Get value from DOL
        - Set PACT to FALSE
3. Convert

- If PACT is FALSE, compute RVAL
    - Set RVAL = VAL
    - Shift RVAL left SHFT bits
- Status=write_mbboDirect

4. If PACT has been changed to TRUE, the device support write output routine has started but has not completed writing the new value. In this case, the processing routine merely returns, leaving PACT TRUE.
5. Check to see if monitors should be invoked.
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - Archive and value change monitors are invoked if MLST is not equal to VAL.
    - Monitors for RVAL and RBV are checked whenever other monitors are invoked.
    - NSEV and NSTA are reset to 0.
6. Scan forward link if necessary, set PACT FALSE, and return.

10. Device Support

10.1. Fields Of Interest To Device Support

Each mbboDirect record must have an associated set of device support routines. The primary responsibility of the device support routines is to obtain a new raw mbbo value whenever write_mbboDirect is called. The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See Chapter 2, *Fields Common to All Record Types* for an explanation of these fields.
DPVT	Device Private	
UDF	VAL Undefined	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
NOBT	Number of Bits	Number of hardware bits accessed. They must be consecutive.
OUT	Output Link	This field is used by the device support routines to locate its output.
RVAL	Raw data value	This is the value to be written to OUT.
RBV	Read Back Value	It is the responsibility of the device support modules to set this field.

Name	Summary	Description
MASK	Mask	This is a mask used to read the hardware. Record support sets the low order NOBT bits. The device support routine can shift the bits. The device support routine should perform the shift in init_record.
SHFT	Shift	This can be set by the device support module at init_record time.

10.2. Device Support Routines

Device support consists of the following routines:

report

```
report(FILE fp, paddr)
```

Not currently used.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine. If MASK is used, it should be shifted if necessary and SHFT given a value.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
     *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

write_mbboDirect

```
write_mbboDirect(precord)
```

This routine must output a new value. It returns the following values:

- 0: Success.
- Other: Error.

10.3. Device Support For Soft Records

This `SOft Channel` module writes the current value of VAL.

If the OUT link type is PV_LINK, then dbCaAddInlink is called by

init_record.

write_mbboDirect calls recGblPutLinkValue to write the current value of VAL.

See *Soft Output, Chapter 3, 3.2*.

Chapter 23:Permissive

1. Introduction

The permissive record is for communication between a server and a client. An example would be a sequence program server and an operator interface client. By using multiple permissive records a sequence program can communicate its current state to the client. The fields in this record fall into the following categories:

scan parameters

client-server parameters

operator display parameters

monitor parameters

2. Scan Parameters

The permissive record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Since the permissive record supports no direct interfaces to hardware, its SCAN field cannot be `I/O Intr`.

3. Client-server Parameters

The client and server communicate through the VAL and watchdog flag (WFLG) fields. At initialization, both fields are set equal to 0, which means OFF. The server sets WFLG equal to ON when it is ready to accept a request. The client monitors WFLG and when WFLG equals 1, the client-server action is performed (a private matter between server and client).

When WFLG is off—when the server is busy—the client program may turn the VAL field from OFF to ON. After the server finishes its task, it will notice that VAL is ON and will turn both WFLG and VAL OFF and performs the requested service.

Note that the when WFLG is ON, the client program *must not* turn VAL to on.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Value Field	USHORT	No	0	Yes	Yes	Yes	Yes
WFLG	Watchdog Flag	USHORT	No	0	Yes	Yes	Yes	Yes

4. Operator Display Parameters

The label field (LABL) contains a string given to it that should describe the record in further detail. In addition to the DESC field. See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LABL	Label	STRING [20]	Yes	Null	Yes	Yes	No	Yes
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

5. Alarm Parameters

The Permissive record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

6. Run-time Parameters

These fields are used to trigger monitors for each field. Monitors for the VAL field are triggered when OVAL, the old value field, does not equal VAL. Likewise, OFLG causes monitors to be invoked for WFLG when WFLG does not equal OLFG.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OVAL	Old Value	USHORT	No	0	Yes	No	No	No
OFLG	Old Flag Value	USHORT	No	0	Yes	No	No	No

7. Record Support Routines

Two record support routines are provided: process, and get_value.

process

process sets UDF to FALSE, triggers monitors on VAL and WFLG when they change, and scans the forward link if necessary.

get_value

get_value fills in struct `valueDes` so that it refers to VAL.

Chapter 24:PID Control

1. Introduction

The PID record is used to maintain a setpoint that affects the output according to the difference between a controlled value and the setpoint. The record reads the controlled value and the setpoint when the record is processed, and then the PID expression computes the result, which is stored in a field (DM) that can be accessed via another record, usually an analog output record.

There are three terms of the PID expression: the proportional, the integral, and the derivative. Using the gain that exists for each of these terms, the user can weight each contribution according to the characteristics of the controlled variable.

The fields in this record fall into several categories:

scan parameters

controlled variable

setpoint parameters

expression parameters

operator display parameters

alarm parameters

monitor parameters

run-time parameters

2. Scan Parameters

The PID record has the standard fields for specifying under what circumstances it will be processed. Note, however, that the integral and derivative parts of the algorithm are affected by the SCAN field, i.e., the scanning algorithm. For Passive, I/O Interrupt, and Event, or a period of 0.1 seconds, the interval is set to 1. For periods of 0.2, 0.5, 1, and 2 seconds, the interval is set to 2, 3, 4, and 5.

These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used.

The minimum delta time field (MDT) is a processing field particular to the PID record. It can be configured by the user or modified at run-time. It contains a floating point value which represents the minimum amount of time between record processing. If the amount of time between the last time the record was processed an the current time is less than MDT, than the record is not processed. If MDT is left at its default value (0), the minimum delta will be equal to one clock tick.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
MDT	Minimum Delta Time	FLOAT	Yes	0	Yes	Yes	No	No

3. Controlled Variable Parameters

The control variable link field (CVL) is used to obtain the value of the controlled process variable, i.e., the value read into the CVAL field. The link must be a database link. If it is not a database link a MAJOR alarm is triggered when the record is processed. See *Address Specification, Chapter 1, 2*, for information on how to specify database links.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
CVL	Controlled Value Location (an input link)	INLINK	Yes	0	No	No	N/A	No
CVAL	Value of controlled variable	FLOAT	No	0	Yes	YesNo	No	No

4. Setpoint Parameters

The setpoint mode select (SMSL) and the setpoint location (STPL) fields determine where the setpoint value is obtained, which is held in the VAL field. The SMSL and STPL fields work just like the OMSL and DOL fields found in many output records (analog output, binary output, etc.).

The SMSL field has two choices: `supervisory` and `closed_loop`. When `supervisory`, the setpoint, i.e., VAL, can be set via database access. When `closed_loop`, the setpoint value is retrieved from the link specified in STPL, which must be a database link. See *Address Specification, Chapter 1, 2*, for information on how to specify database links. STML can also be a constant in which case VAL is set equal to the constant value when the record is initialized.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
STPL	Setpoint Location (an input link)	INLINK	Yes	0	No	No	N/A	No
SMSL	Setpoint Mode select.	GBLCHOICE	Yes	0	Yes	Yes	No	No
VAL	Setpoint value	FLOAT	No	0	Yes	Yes	Yes	Yes

5. Expression Parameters

The discrete from of the PID expression is as follows:

$$M(n) \;=\; KP \times E(n) + KI \times Sumi\,(E(i) \times dT(n)) \;+ KD \times \left(\frac{E(n) - E(n\text{-}1)}{dT(n)}\right) + Mr$$

where

M(n) = value of manipulated variable at *n*th instant.

KP = Proportional gain

KI = Integral gain

KD = Derivative gain

E(n) = Error nth sampling instant

SUMi = Sum from i=0 to i=n

dT(n) = Time difference between n-1 instance and nth instance

Mr = Midrange adjustment.

Taking the first difference between instances yields the following equation:

$$delM(n) \;=\; KP \times \;(E(n) - E(n\text{-}1)) \;+ KI \times E(i) \times dT(n) + KD \times \frac{E(n) - E(n\text{-}1)}{dT(n)} - \frac{(E(n) - E(n\text{-}1))}{dT(n\text{-}1)}$$

The terms KP, KI, and KD are the only terms configured by the user. These terms represent the gain for the proportional (KP), integral (KI) and the derivative (KD). A field exists for each of these terms (the KP, KI, and KD fields). KP should be configured according to the characteristics of the controlled variable; KI should be set equal to the number of times that the integral contribution repeats the proportional contribution; and KD should equal the number of minutes until the derivative contribution repeats the proportional contribution.

The PID record calculates the other terms of the expression:

E_n — Error at nth sampling instant, where the Error equals the setpoint minus the value of the controlled variable (VAL - CVAL).

delM(n) — This is the end result of the PID expression—the change in the manipulated value. The DM field holds this value.

dT(n) — This is the time difference between n and n-1, between current and last samplings. It is stored in the DE field.

The final result of the expression `delM(n)` and the other parts of the expression are held in several different run-time fields, which are not configurable prior to run-time, nor modifiable at run-time, but can be accessed so that their values can be used. See Section 9, *Run-time Parameters*, in this chapter for more information on those fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
KP	Proportional Gain	FLOAT	Yes	0	Yes	Yes	No	No
KI	Integral Gain, in repeats per minute.	FLOAT	Yes	0	Yes	Yes	No	No
KD	Derivative Gain, in repeats per minute	FLOAT	Yes	0	Yes	Yes	No	No

6. Operator Display Parameters

These parameters are used to present meaningful data to the operator. They display the setpoint (VAL), the controlled variable (CVAL), the change in manipulated value (DM), and other fields of the PID, either textually or graphically.

EGU is a string of up to 16 characters describing the units of PID's manipulated values measures. It is retrieved by the `get_units` record support routine. It must be configured by the user if at all.

The HOPR and LOPR fields set the upper and lower display limits for the VAL, HIHI, HIGH, LOW, LOLO, VAL, and CVAL fields.

The PREC field determines the floating point precision with which to display VAL and CVAL. It is used whenever the `get_precision` record support routine is called.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	null	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

7. Alarm Parameters

The possible alarm conditions for PID are the SCAN alarm, limit alarms, and an alarm that is triggered when CVL is not a database link. The SCAN and "CVL" alarms are called by the record routines and are always of MAJOR severity.

The limit alarms trigger alarms on the VAL field. They are configured by the user in the HIHI, LOLO, HIGH, and LOW fields using floating point values. For each of these fields, there is a corresponding severity field which can be either NO ALARM, MINOR, or MAJOR. See *Alarm Specification, Chapter 1, 4*, for a complete explanation of alarms and these fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HIHI	Hihi Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOLO	High Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HIGH	Low Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOW	Lolo Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HHSV	Hihi Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LLSV	High Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HSV	Low Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LSV	Lolo Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HYST	Alarm Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

8. Monitor Parameters

These parameters are used to determine when to send monitors placed on the VAL field. These fields contain values configured by the user. The monitors are sent when the VAL field exceeds the last monitored field (MLST, ALST, and LALM) by the appropriate delta. Otherwise, value change monitors are not called. If these fields have a value of zero, everytime the value changes, a monitor will be triggered; if they have a value of -1, everytime the record is processed, monitors are triggered. The ADEL field is the delta used for archive monitors, and the MDEL field is the delta for all other types of monitors.

The ODEL field specifies the hysteresis factor for the DM field, the field which holds the manipulated value. Unless the current value of DM is greater than the amount which the user specifies in this field, no monitors will be invoked. If zero, anytime DM is greater than zero, a monitor is triggered. If -1, each time the record is processed, a monitor is triggered. Note that when monitors are called for DM, they are also called for the following fields which comprise the PID expression: P, I, D, CT, DT, ERR, and DERR.

See *Monitor Specification, Chapter 1, 5*, for a complete explanation of monitors.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ADEL	Archive Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
MDEL	Monitor, i.e. value change, Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
ODEL	Output deadband	FLOAT	Yes	0	Yes	Yes	No	No

9. Run-time Parameters

These parameters are used by the run-time code for processing the PID. They are not configurable prior to run-time. They represent the current state of the PID and/or the terms of the PID expression. Many of them are used to process the PID record more efficiently.

The DM field contains the change in manipulated value, the result of the PID expression (delM(n)). It is an increment which will usually be accessed by the desired output link (DOL) of an analog output record. Since its value represents an increment, the OIF field of the analog output record should be set to `Incremental`. There is no output field with which the PID record can itself send output.

The P, I, D, CT, DT, ERR, and DERR fields are used to calculate the PID expression. They are not configurable prior to, nor modifiable during, run-time, but may be of interest when fine tuning the gains of each contribution (KP, KI, KD). The following lists the fields as they relate to the expression:

ERR	`En` Error at current sampling (VAL-CVAL).
DERR	`En - En-1` Difference between current error and error at last sampling.
P	This field corresponds to `KP * (En - (En-1))`
I	This field corresponds to `En * dTn * KI`
D	This field corresponds to `Kp * Kd * (err/dt(n) - derr/dt(n-1))`

The LALM, ALST, and MLST fields are used by record processing to implement the monitors for this record. These fields hold the values for the VAL field from the last time the record was processed. When the record is processed again the difference between these fields and current value of VAL exceeds the appropriate delta (MDEL for instance), then the appropriate monitors are triggered.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
DM	Change in Manipulated Value	FLOAT	No	0	Yes	No	Yes	No
ODM	Old DM.	FLOAT	No	0	Yes	No	Yes	No
P	Proportional contribution to DM.	FLOAT	No	0	Yes	No	Yes	No
I	Integral contribution to DM.	FLOAT	No	0	Yes	No	Yes	No
D	Derivative contribution to DM.	FLOAT	No	0	Yes	No	Yes	No
CT	Clock ticks when previous process occurred.	ULONG	No	0	Yes	No	Yes	No
DT	Time difference in seconds between processing steps.	FLOAT	No	0	Yes	No	Yes	No
ERR	Current error (VAL - CVAL).	FLOAT	No	0	Yes	No	Yes	No
DERR	Delta Error	FLOAT	No	0	Yes	No	Yes	No
LALM	Value when last monitors for alarm were triggered	FLOAT	No	0	Yes	No	No	No
ALST	Value when last monitors for archiver were triggered	FLOAT	No	0	Yes	No	No	No
MLST	Value when last monitors for value changes were triggered	FLOAT	No	0	Yes	No	No	No

10. Record Support Routines

init_record

If STPL is a constant link, initialize VAL with it’s value and set UDF to false.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_units

Retrieves EGU.

get_precision

Retrieves PREC.

get_graphic_double

Sets the following values:

upper_disp_limit = hopr

lower_disp_limit = lopr

get_control_double

Sets the following values

upper_ctrl_limit = hopr

lower_ctrl_limit = lopr

get_alarm_double

Sets the following values:

upper_alarm_limit = hihi

upper_warning_limit = high

lower_warning_limit = low

lower_alarm_limit = lolo

11. Record Processing

Routine process implements the following algorithm:

1. If CVL is not a database link a major alarm is declared and the algorithm completes.
2. The current value of CVAL is obtained from CVL.

3. If STPL is a database link and SMSL is CLOSED_LOOP then VAL is obtained from STPL and UDF is set to false.
4. The time difference since the last time step is calculated. If it is less than MDT or if no ticks have occurred since the last time the algorithm was executed, process just completes without raising any alarms, checking monitors, or scanning the forward link.
5. The new values of P, I, D, OUT, CT, DT, ERR, and DERR are computed.
6. Check alarms. This routine checks to see if the new VAL causes the alarm status and severity to change. If so NSEV and NSTA and LALM are set. It also honors the alarm hysteresis factor (HYST). Thus the value must change by at least HYST before the alarm status and severity changes.
7. Checks to see if monitors should be invoked:
    - Alarm Monitors are invoked ADEL and MDEL conditions are met.

Archive and value change monitors are invoked if ODEL conditions are met. If monitors are triggered from DM, they are also triggered for P, I, D, CT, DT, E

Chapter 25:pulseCounter

1. Introduction

The normal use for the pulseCounter record type is to record counts and write them to a device. Its fields fall into the following categories:

scan parameters

setup parameters

write parameters

operator display parameters

run-time parameters

2. Scan Parameters

The pulse counter record has the standard fields for specifying under what circumstances it will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Setup Parameters

These parameters control the characteristics of the counter pulse. Normally, the user only needs to configure the gate type (GTYP) field. This field has two choices, `Hardware` or `Software`. If hardware is chosen, then the HGV field controls gating. The device support routine should set the HGV field.

If GTYP is configured as software, then gating control is determined by the soft gate location (SGL) and soft gate value (SGV) fields. SGL must be a database link; SGV is read from that link. SGV can have two possible values, `Active` or `Inactive`. `Active` causes the command routine (see below section on run-time parameters) to start counting, while `Inactive` causes it to stop counting. See *Address Specification, Chapter 1, 2*, for information on how to specify database links.

The device support routines should also set the counter size (CSIZ), counter edge (CNTE), and count source (CNTS) fields. Nevertheless, these fields can be configured using a configuration tool. CSIZ can be 16 bit or 32 bit. With 32 bit, two counters are used. The CNTE field determines whether counting occurs on the rising or falling edge of the source signal. It's choices are `Rising Edge` or `Falling Edge`. The CNTS determines the count source during setup.

Once again, most of these fields are set by the device support routines.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
GTYP	Gate Type	RECCHOICE	Yes	0	Yes	Yes	No	No
SGL	Soft Gate Location (Input Link)	INLINK	Yes	0	No	No	N/A	No
SGV	Soft Gate Value	RECCHOICE	Yes	0	Yes	Yes	No	No
HGV	Hardware Gate Value	SHORT	Yes	0	Yes	Yes	No	No
CSIZ	Counter Size	RECCHOICE	Yes	1	Yes	Yes	No	No
CNTE	Count Edge	RECCHOICE	Yes	0	Yes	Yes	No	No
CNTS	Count Source	SHORT	Yes	0	Yes	Yes	No	No

4. Write Parameters

The output link (OUT) specifies where the pulse counter write its output, i.e., the current, recorded value of the counter (VAL). If the record writes its a value to a device, the OUT output link must specify the address of the I/O card, and the DTYP field must contain the name of the appropriate device support module. Be aware that the address format differs according to the I/O bus used. See *Address Specification, Chapter 1, 2*, for information on the format of hardware addresses. You can see a list of the device support modules currently supported at the user's local site by using the dbst utility in R3.13.

If the record uses soft device support, then it can be a database or channel access link, or a constant. If it's a constant, no output can be written.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OUT	Output Link	OUTLINK	Yes	0	No	No	N/A	No
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	No

5. Operator Display Parameters

These parameters are used to present meaningful data to the operator. The display the value and other parameters of the pulse counter either textually or graphically. The operating range fields (HOPR and LOPR) for the pulse counter record apply only to the VAL field. They are retrieved when the `get_graphic_double` and `get_control_double` record support routines are called. See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HOPR	High Operating Range	FLOAT	Yes	4.3e+9	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

The pulse counter record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

7. Run-time parameters

These parameters are used by the record itself and the device support routines. They are not configurable prior to run-time, and only the CMD and VAL fields are modifiable by a user at run-time.

The CMD field controls the command routines that control the counter. It has five commands:

`Read` Read the current value of the counter

`Clear` Stop and clear the counter, i.e., reset it to zero. Since the Stop command is also invoked, the Start command must be issued to start the counter after this command is invoked.

`Start` Start counting

`Stop` Stop counting. This does not reset the counter.

`Setup` Do not begin counting until Start command is issued.

The VAL field holds the current number of recorded pulses. It is normally set by the read command.

The record support routines and device support routines sometimes use the SCMD field to temporarily store the currently active CMD, after which they change the current, active command in the CMD field. When they are done, they will set the command back to the saved command.

The OSGV field is used to implement monitors for the SGV field.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
CMD	Command	RECCHOICE	No	0	Yes	Yes	Yes	Yes
SCMD	Save Command	USHORT	No	0	Yes	No	No	No
CPTR	Callback	ULONG	No	0	Yes	No	No	No
VAL	Counter Value	ULONG	No	0	Yes	Yes	Yes	No
OSGV	Old Soft Gate Value	SHORT	No	0	Yes	No	No	No

8. Record Support Routines

init_record

This routine next checks to see that device support is available. If it does not exist, an error message is issued and processing is terminated.

If SGL is a constant and GTYP is software, then SGV is initialized with its value. If SGL type is PV_LINK a channel access link is created.

Device support is then checked to see if cmd_pc is defined.

If device support includes init_record, it is called.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_graphic_double

Sets the upper display and lower display limits for a field. If the field is VAL the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_control_double

Sets the upper control and the lower control limits for a field. If the field is VAL the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

9. Record Processing

The routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. If SGL is DB_LINK and GTYP is Software, get SGV from SGL. If SGV has changed, save the CMD value, call the command routine with START if SGV =0 or with STOP if SGV is 1, reset the command to the saved value, and set alarms if return status not zero. If the device is not done (PACT TRUE), then issue a callback request for this record to process and return
3. If CMD is not READ, call command routine and set CMD to READ. If the device is not done (PACT TRUE), then issue a callback request for this record to process again and return.
4. Call command routine. If device support set PACT to TRUE, then return.
5. Check to see if monitors should be invoked.
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - Archive and value change monitors on CMD are invoked if values have changed.
    - NSEV and NSTA are reset to 0.
6. Scan forward link if necessary, set PACT FALSE, and return.

10. Device Support

10.1. Fields Of Interest To Device Support

Each record must have an associated set of device support routines. The primary responsibility of the device support routines is to issue commands to the output device. The device support routines are primarily interested in the following fields:

Name	Summary	Description
CSIZ	Counter size	This will determine to a 16 bit or 32 bit count is to be used. With 32 bit, two counters are used.
CMD	Command	The device support routine is responsible for processing the commands READ, CLEAR, START, STOP, and SETUP.
GTYP,IGV	Gate Type	If GTYP is internal, device support is responsible for using IGV to determine gating control.
CNTE	Count Edge	This field is used by the device support routines to force counting on leading or falling edge of signal.

Name	Summary	Description
CNTS	Count Source	Device support must use CNTS to set count source during setup.

10.2. Device Support Routines

Device support consists of the following routines:

report

```
report()
```

This routine is optional. If provided, it prints a report of all device modules.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
      *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

cmd_pc

```
cmd_pc(precord)
```

This routine issues commands to the output device. It returns the following values:

0: Success.

Other: Error.

Chapter 26:pulseDelay

1. Introduction

The normal use for the pulseDelay record type is to generate pulses to be written to a device. Its fields fall into the following categories:

scan parameters

trigger parameters

pulse parameters

output parameters

operator display parameters

run-time parameters

2. Scan Parameters

The pulse delay record has the standard fields for specifying under what circumstances it will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Trigger Parameters

These fields determine the source for the pulse triggers. Since the pulse delay record is a kind of output record, these fields resemble the desired output parameters of, for example, the analog output record.

The trigger type (TTYP) field determines whether the pulse trigger comes from an external or internal source. Or, differently put, from a hardware source or from a soft trigger source. When TTYP is set to `Hardware`, the HTS field becomes the trigger source for the record. The hard device support sets the HTS. When hardware is chosen, the OUT output link must specify a hardware address, and the DTYP field must specify the appropriate device support module.

When software is specified in the TTYP field, the record uses the SGV field to generate pulses. The SGV field generates a pulse when its value is `Active`, and generates no pulse when `Inactive`. It's value can be manipulated by dbPuts at run-time, or else a value for STV can be retrieved from the soft trigger location (STL) field, an input link which must be a database link if STV is used.

The GATE field can either enable or disable the generation of pulses. When GATE is 0, no pulses are triggered. When GATE is 1, pulses are triggered according to the record's configuration. A value for GATE can be read from the GLNK field, an input link which can be a constant, database link, or a channel access link. When a constant, GATE is initialized to the constant value. Otherwise, a value for GATE is fetched from the location specified in GLNK each time the record is processed.

The CEDG, CTYP, ECS, and ECR field may not have significance for all device support modules. For the modules they apply to, they control the timing of pulses. The CEDG field determines whether clock timing occurs on the rising edge or falling edge of a signal. It has two choices: `Rising Edge` or `Falling Edge`. The CTYP field determines whether the timing is controlled externally or internally and has two choices: `Internal` and `External`. The ECS and ECR fields have significance only if CTYP specifies `External`.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
TTYP	Trigger Type (hardware/Software)	RECCHOICE	Yes	0	Yes	Yes	No	No
HTS	Hardware Trigger	ENUM	Yes	0	Yes	Yes	No	Yes
STL	Soft Trigger Location (input link)	INLINK	Yes	0	No	No	N/A	No
STV	Soft Trigger Value	RECCHOICE	Yes	0	Yes	Yes	No	Yes
GATE	Gate for enable/disable of Pulse Generation	RECCHOICE	Yes	1	Yes	Yes	No	Yes
GLNK	Gate Location	INLINK	Yes	0	Yes	No	N/A	No
CTYP	Clock Type	RECCHOICE	Yes	0	Yes	Yes	No	No
CEDG	Clock Signal Edge	RECCHOICE	Yes	0	Yes	Yes	No	No
ECS	External Clock Source	SHORT	Yes	0	Yes	Yes	No	No
ECR	External Clock Rate, in Hz	DOUBLE	Yes	0	Yes	Yes	No	No

4. Pulse Parameters

These fields determine the characteristics of the pulse that is generated by the record. The pulse delay (DLY) field is the most important of these fields. In it, the user specifies the time delay, from the trigger edge until the generation of the pulse. In the time units (UNIT) field, the user specifies the units of time that the delay should be in. The UNIT field can specify `Seconds`, `Milliseconds`, `Microseconds`, `Nanoseconds`, or `Picoseconds`.

The user specifies the time duration of the pulse in the pulse width (WIDE) field, which also uses the UNIT field for its time units.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
UNIT	Time units	RECCHOICE	Yes	0	Yes	Yes	No	No
DLY	Pulse Delay, in UNITS of time	DOUBLE	Yes	0	Yes	Yes	Yes	Yes
WIDE	Pulse Width, in UNITS of time	DOUBLE	Yes	0	Yes	Yes	Yes	Yes

5. Operator Display Parameters

These parameters are used to display the delay value (DLY)and other parameters of the pulse delay either textually or graphically.

The HOPR and LOPR fields set the upper and lower display limits for the VAL, DLY, ODLY, WIDE, and OWID fields. Both the `get_graphic_double` and `get_control_double` record support routines retrieve these fields.

The PREC field determines the floating point precision with which to display DLY only. It is used whenever the `get_precision` record support routine is called.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

The pulse delay record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

7. Run-time Parameters

These fields are used by the record for processing and to implement monitors for some of the pulse fields.

The old delay (ODLY) and old width (OWID) fields are used to implement monitors for the DLY and WIDE fields, respectively. If, for instance, the current value of the DLY field differs from the value held in ODLY, monitors are triggered for the DLY field.

The PFLD indicates which of the following fields have changes since the record was last processed: DLY, WIDE, STV, GATE, or HTS. Some devices use PFLD so that adjustments can be made when any of these fields are changed.

The VAl field indicates whether a pulse has been generated since the last time the record was processed. It is `ACTIVE` if YES, `INACTIVE` if NO.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ODLY	Old Delay	DOUBLE	No	0	Yes	No	No	No
OWID	Old Width	DOUBLE	No	0	Yes	No	No	No
PFLD	Processing Field	USHORT	No	0	Yes	No	No	No
VAL	Value	RECCHOICE	No	0	Yes	Yes	Yes	No
LLOW	Low Logic Level	RECCHOICE	Yes	0	Yes	Yes	No	No

8. Record Support Routines

init_record

This routine first checks that device support is available. Device support is then checked to see if write_pd is defined.

Next this routine initializes STV with the value of STL if STL type is CONSTANT link or creates a channel access link if STL type is PV_LINK.

GATE is likewise initialized if GLNK is CONSTANT or PV_LINK.

If device support includes init_record, it is called.

process

See next section.

special

Sets the PFLD field to indicate if write to DLY, STV, GATE or HTS field caused processing of the record.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_precision

Retrieves PREC.

get_graphic_double

Sets the upper display and lower display limits for a field. If the field is VAL, DLY, ODLY, WIDE or OWID the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_control_double

Sets the upper control and the lower control limits for a field. If the field is VAL, DLY, ODLY, WIDE or OWID the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

9. Record Processing

Routine process implements the following algorithm:

Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.

1. The values for STV and GATE are then fetched.
2. Call write_pd routine.
3. PFLD is reset to zero.
4. If device support set PACT to TRUE, then return.
5. Set UDF to FALSE.
6. Check to see if monitors should be invoked.
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - Archive and value change monitors on DLY and WIDE are invoked if values have changed.
    - NSEV and NSTA are reset to 0.
7. Scan forward link if necessary, set PACT FALSE, and return.

10. Device Support

10.1. Fields Of Interest To Device Support

Each record must have an associated set of device support routines. The primary responsibility of the device support routines is to issue commands to the output device. The device support routines are primarily interested in the following fields:

Name	Summary	Description
OUT	Output Link	This field is used by the device support routines to locate its output.
WIDE	Pulse Width	Device support must use WIDE for pulse width
DLY	Pulse Delay	Device support must use DLY for the delay after trigger edge until beginning of pulse.
LLOW	Low Logic Level	Device support must use to determine logic low level.
UNIT	Time Units	All values that refer to time measure will be in this time unit.
VAL	Value	This field will contain a 1 if a trigger occurred since the last time the record was processed if the device supports it.
PFLD	Processing Field	This field is used by some devices to indicate if the record was scanned to adjust certain fields such as delay or trigger source. If the device has a destructive read, then changes to these types of fields could write to the device instead of a read and a write.
TTYP	Trigger Type	This field is used by the device support routines to force triggering on leading or falling edge of signal if the specified device supports it.
HTS	Hardware Trigger Source	This field will be used to set the hardware trigger source if the device supports it.
STV	Soft Trigger Source	This field will be used for software to trigger an output delayed pulse if the device supports it.

Name	Summary	Description
CEDG	Clock Signal Edge	This field is used by the device support routines to force clock timing on leading or falling edge of signal.
CTYP	Clock Type	If CTYP is external, device support is responsible for using ECR for the clock rate and if CTYP is internal, ECS is the clock source.
ECS	External Clock Source	
ECR	ExternalClockRate	

10.2. Device Support Routines

Device support consists of the following routines:

report

```
report()
```

This routine is optional. If provided, it prints a report of all device modules.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
     *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

write_pd

```
write_pd(precord)
```

This routine issues commands to the output device.

Chapter 27:pulseTrain

1. Introduction

The normal use for the pulseTrain record type is to generate a pulse train for output. Its fields fall into the following categories:

scan parameters

trigger parameters

pulse parameters

output parameters

operator display parameters

run-time parameters

2. Scan Parameters

The pulseTrain record has the standard fields for specifying under what circumstances the it will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Trigger Parameters

These fields determine the source of the trigger for each train. At least the gate type (GTYP) field must be configured by the user. It can be either `Hardware` or `Software`. If `Hardware`, then the triggers are device dependent, and the HGV field determines gating control, HGV being device dependent.

If the user configures GTYP to be software, then the soft gate value (SGV) field provides gating control. This field has the same possible values as HGV, `Active` and `Inactive`, triggering a pulse when `Active`, or not triggering a pule when `Inactive`. The SGV value can be retrieved from the location specified in the Soft Gate Location (SGL) field, which must be a database link.

The CEDG, CTYP, ECS, and ECR field may not have significance for all device support modules. For the modules they apply to, they control the timing of pulses. The CEDG field determines whether clock timing occurs on the rising edge or falling edge of a signal. It has two choices: `Rising Edge` or `Falling Edge`. The CTYP field determines whether the timing is controlled externally or internally and has two choices: `Internal` and `External`. The ECS and ECR fields have significance only if CTYP specifies `External`.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
GTYP	Gate Type	RECCHOICE	Yes	0	Yes	Yes	No	No
HGV	Hardware Gate Value	SHORT	Yes	0	Yes	Yes	No	No
SGL	Soft Gate Location (Input Link)	INLINK	Yes	0	No	No	N/A	No
SGV	Soft Gate Value	RECCHOICE	Yes	0	Yes	Yes	No	No
CTYP	Clock Type	RECCHOICE	Yes	0	Yes	Yes	No	No
CEDG	Clock Signal Edge	RECCHOICE	Yes	0	Yes	Yes	No	No
ECS	External Clock Source	SHORT	Yes	0	Yes	Yes	No	No
ECR	External Clock Rate, in Hz	DOUBLE	Yes	0	Yes	Yes	No	No

4. Pulse Parameters

These parameters characterize the pulse train. The pulse train is characterized by the time duration of the entire train, which the user specifies in the Period (PER) field, and the percentage of that time when the signal is high, which the user specifies in the Duty Cycle (DCY) field. In the UNIT field, the user must specify the units of time which the time period is in. This field can be either `Seconds`, `Milliseconds`, `Microseconds`, `Nanoseconds`, or `Picoseconds`.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
UNIT	Units of time	RECCHOICE	Yes	0	Yes	Yes	No	No
PER	Period, in UNITs	DOUBLE	Yes	0	Yes	Yes	Yes	No
DCY	Duty Cycle, percent	DOUBLE	Yes	0	Yes	Yes	Yes	No

5. Output Parameters

The OUT field contains the location where the pulse train record sends the actual pulse. If the pulseTrain uses hardware device support, it must specify the address of the I/O card, and the DTYP field must specify the appropriate device support module. Be aware that the address format differs according to the I/O bus used. See *Address Specification, Chapter 1, 2*, for information on the format of hardware addresses. You can see a list of the device support modules currently supported at the user's local site by using the dbst utility in R3.13.

For records that use soft device support, OUT can be a constant, a database link or a channel access link, though if it's a constant, no output will be written. See *Address Specification, Chapter 1, 2*, for information on how to specify database links.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OUT	Output Link	OUTLINK	Yes	0	No	No	N/A	No
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	No

6. Operator Display Parameters

These parameters are used to present meaningful data to the operator. They display the value and other parameters of the pulse train either textually or graphically. The operating range fields (HOPR and LOPR) for the pulse train record apply to the VAL, PER, or OPER fields. They are retrieved when the `get_graphic_double` and `get_control_double` record support routines are called for that field. The PREC field is used when the `get_precision` record routine is called. It is used only for the value in the VAL field.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HOPR	High Operating Range	FLOAT	Yes	4.3e+9	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

7. Alarm Parameters

The pulse train record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

8. Run-time Parameters

The record uses these fields for processing and to implement monitors. The VAL field is not used.

The Old Period (OPER), Old Duty Cycle (ODCY), and Old Soft Gate Value (OSGV) fields are used to implement monitors for the PER, DCY, and SGV fields, respectively. When the current value doesn't equal the old value, monitors are called for that field.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OPER	Old Period, in UNITS	DOUBLE	No	0	Yes	No	Yes	Yes
ODCY	Old Duty Cycle, percent	DOUBLE	No	0	Yes	No	Yes	Yes
OSGV	Old Soft Gate Value	SHORT	No	0	Yes	No	No	No
VAL	Value	SHORT	No	0	Yes	Yes	Yes	Yes
LLOW	Low Logic Level	RECCHOICE	Yes	0	Yes	Yes	No	No

9. Record Support Routines

init_record

This routine first checks that device support is available. If SGL is a constant then HGV is initialized with its value or a channel access link is created if SGL type is PV_LINK.

Device support is then checked to see if write_pt is defined.

If device support includes init_record, it is called.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_precision

Retrieves PREC.

get_graphic_double

Sets the upper display and lower display limits for a field. If the field is VAL, PER, or OPER the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_control_double

Sets the upper control and the lower control limits for a field. If the field is VAL or PER the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

10. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. If SGL is DB_LINK and GTYP is Software, get SGV from SGL. If SGV has changed, save the duty cycle DCY value, call the write_pt routine with duty cycle =0, reset the duty cycle to the saved value, and set alarms if return status not zero. Then set the old soft gate value OSGV to SGV.
3. Call write_pt routine. If device support set PACT to TRUE, then return.
4. Set UDF to FALSE.
5. Check to see if monitors should be invoked.
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - Archive and value change monitors on PER and DCY are invoked if values have changed.
    - NSEV and NSTA are reset to 0.
6. Scan forward link if necessary, set PACT FALSE, and return.

11. Device Support

11.1. Fields Of Interest To Device Support

Each record must have an associated set of device support routines. The primary responsibility of the device support routines is to issue commands to the output device. The device support routines are primarily interested in the following fields:

Name	Summary	Description
UNIT	Units of time	This field will be used to identify the time units used for time fields.
OUT	Output Link	This field is used by the device support routines to locate its output.
PER	Period, in UNITs	Device support must use PER for pulse period.

Name	Summary	Description
DCY	Duty Cycle, percent	Device support must use DCY for the percent of time the signal is high.
LLOW	Low Logic Level	Device support must use to determine logic low level.
CEDG	Clock Signal Edge	This field is used by the device support routines to force counting on leading or falling edge of signal.
GTYP	Gate Type	Device support is responsible for using IGV to determine gating control if GTYP is internal, or SGV if GTYP is external.
SGV	Soft Gate Value	
CTYP	Clock Type	If CTYP is external, device support is responsible for using ECR for the clock rate and if CTYP is internal, ECS is the clock source.
ECS	External Cocksurely	
ECR	External Clock Rate, in Hz	

11.2. Device Support Routines

Device support consists of the following routines:

report

```
report()
```

This routine is optional. If provided, it prints a report of all device modules.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
      *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

write_pt

```
write_pt(precord)
```

This routine issues commands to the output device. It returns the following values:

0: Success.

Other: Error.

11.3. Soft Device Support

The `Soft Channel` device support module writes the current value of VAL.

Chapter 28:scan

Ned D. Arnold
Advanced Photon Source
Argonne National Laboratory

1. Introduction

The basic function of a scan record is to move *positioners* through a series of steps and record *detector* data at each of the positions, the whole sequence being referred to as a *scan*. Once the scan parameters are properly initialized, the scan record coordinates the entire scan and notifies any interested clients when the scan is complete. The data is stored in arrays within the record rather than collected point to point by an external application program. This allows for much faster scans than those coordinated from an external application program on a point to point basis.

A single scan record supports a one dimensional scan. It is also possible to link scan records together to define multi-dimensional scans in a complex configuration.

Each scan record can control up to four positioners and acquire data from up to four process variables (typically detector data or measured positions of devices) during a scan. Two additional output variables can be defined to trigger other process variables (usually *detectors*) between the positioning phase and the data acquisition phase. These outputs will be referred to as *detector triggers*.

Although the typical use of a scan record is to move positioners and record detector data at each position, it can also be used for other applications. Any controllable device can be scanned through incremental values while recording data from any other process variables. For example, one of the positioner process variables could be used to vary the gain of a detector or the speed of a motor during a scan. Another example would be to use the scan record to vary several power supplies and record the beam position at each value of the supplies. In this context, the scan record becomes a general purpose “Vary w, x, y, z and record a, b, c, d” record. Therefore, throughout this document the word *positioner* and *controller* will be used synonymously. When referring to the data being recorded at each point, the word *detector* will be used.

All of the process variable names used to identify positioners, detectors, and detector triggers are specified using *reassignable links*. This allows a scan to be configured on the fly.

Scan parameters, including the names of controllers and detectors, can be saved and restored using the BURT.

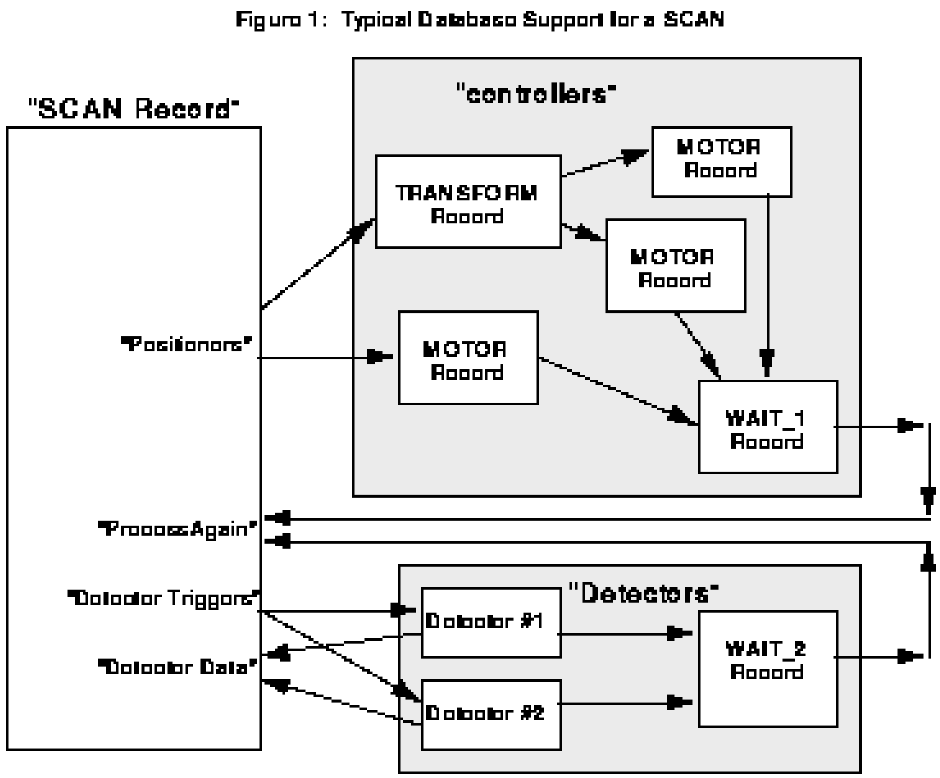
NOTE: In this version, the PVs used in the reassignable fields must reside in the same IOC.

1.1. A Simple Single Dimensional Scan

The simplest database configuration for using a scan record is shown in Figure 1. A thorough understanding of the operation of this configuration will allow more complex scans to be developed easily.

In Figure 1, when the scan is initiated, the scan record commands several positioners (TRANSFORM record and MOTOR record) to move to their starting positions. The WAIT_1 record detects when all movement is complete and forces the SCAN record to process again. The SCAN record realizes that the positioning is complete and thus triggers the Detectors. The WAIT_2 record detects when data is valid and forces the SCAN record to process yet again. The SCAN record will then read the Detector Data and command the positioners to their next step. This will continue until the SCAN record has completed the appropriate number of steps. At the end of the scan, the SCAN record contains data arrays for each of the *detectors*, as well as arrays that contain the positions to which each controller was commanded at each point. A simple x-y plot using this data will provide the detector data vs. position results.

Figure 10 Typical Database Support for a SCAN



1.2. Two Dimensional Scanning

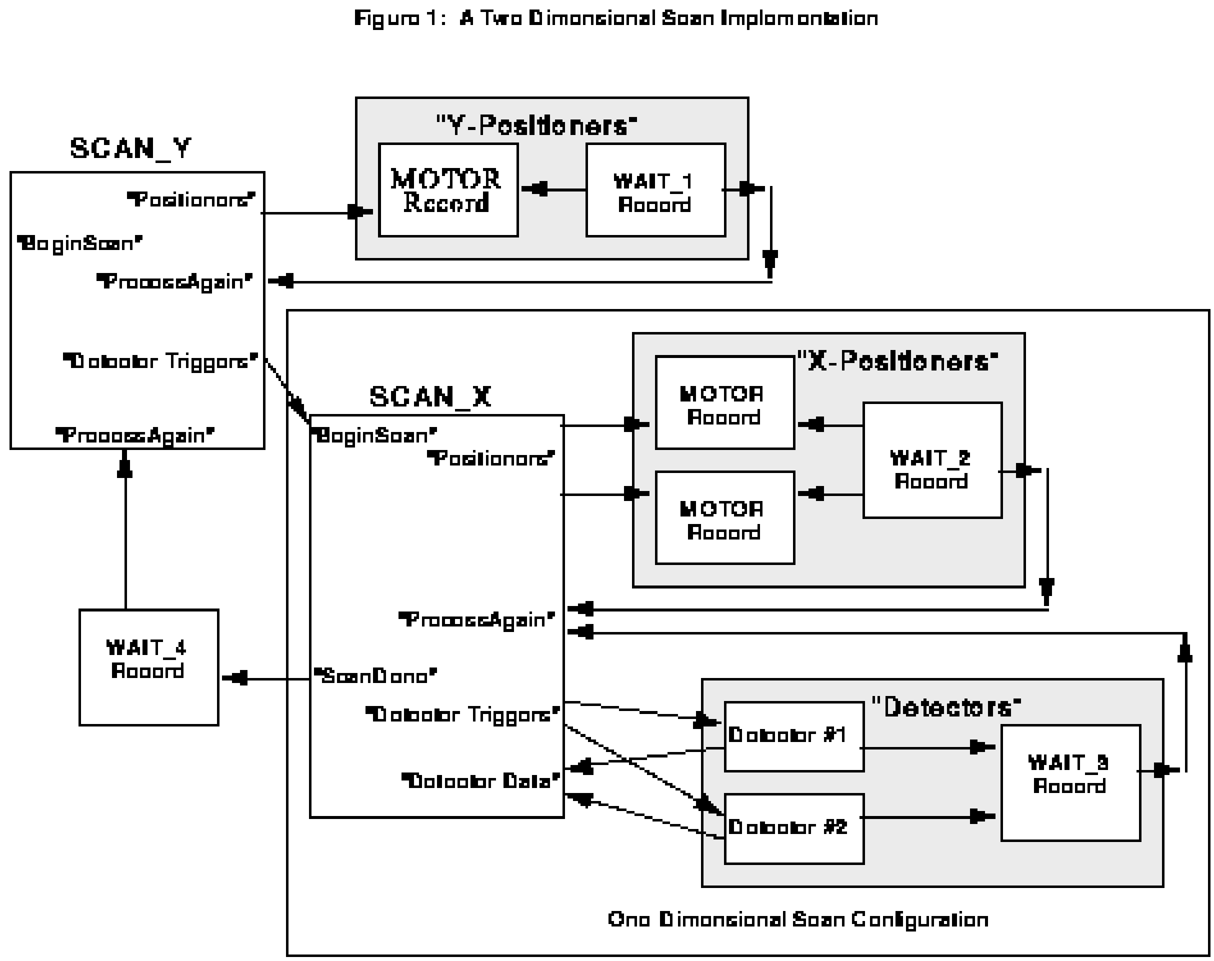
Figure 2 illustrates using two scan records to accomplish a two dimensional scan. The SCAN_X record controls the positioners for the X axis, while the SCAN_Y record controls the Y positioner.

To initiate a scan, the SCAN_Y record is commanded to begin. It commands its *positioners* to the specified starting point. The WAIT_1 record detects when all motors are stopped and forces the SCAN_Y record to process again. The SCAN_Y record will now write to its *Detector Trigger*, which in this case begins a scan of the SCAN_X record. The SCAN_X record will now go through its entire programmed scan, acquiring data from the detectors at each point.

When the SCAN_X record is complete, the WAIT_4 record will force the processing of SCAN_Y, which will increment the position of the y-controller and initiate the x scan once again.

This approach to configuring a two dimensional scan is very flexible. Note that to test the x scan, one could write to the `begin scan` field of SCAN_X which would perform an entire x scan. Although the SCAN_Y record would get processed after the x scan was complete (via the WAIT_4 record), nothing would happen because it was not in the middle of a scan. In addition, any of the motors can be moved individually when a scan is not in process without any unexpected behavior (detectors would not be triggered unless the operator did it deliberately). One could even build a three dimensional scan by adding an additional scan record that initiates the y-scan after positioning a z-controller.

Figure 11 A Two Dimensional Scan



2. Scan Parameters

Several options are available to control the execution of a scan. All parameters must be properly configured prior to initiating the scan.

2.1. Positioner Parameters

Each scan record may control up to four *positioners* that are commanded to a new *desired position* after collecting data at each point. The positioners are defined by typing in an ASCII string that represents the process variable name of the controller.

There are three modes for determining the desired value for the positioner. The desired mode is specified in the P1SM-P4SM fields: `Linear`, `Table`, and `On-The-Fly`. If a positioner is specified as `Linear`, its desired value is determined by using parameters such as start position, step increment, number of points, and end position (which are explained below). If a positioner is specified as `Table`, its next position is found in an array that has been loaded into the record prior to initiating a scan. If the positioner is specified as `On-The-Fly`, it is commanded to the end position after the first data point is collected and not changed again for the duration of the scan.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
P1SM	Positioner 1 Step Mode	RECCHOICE	Yes	0	Yes	Yes	No	No
P2SM	Positioner 2 Step Mod	RECCHOICE	Yes	0	Yes	Yes	No	No
P3SM	Positioner 3 Step Mode	RECCHOICE	Yes	0	Yes	Yes	No	No
P4SM	Positioner 4 Step Mode	RECCHOICE	Yes	0	Yes	Yes	No	No

2.2. Linear Mode

If a positioner's step mode field (P1SM) specifies `Linear`, a scan can be fully defined by three parameters, e.g., the start position (P1SP), the step increment (P1SI), and the number of data points (NPTS). A scan involving *N* controllers is defined by merely 2*N*+1 parameters, since NPTS applies to all positioners. For the convenience of interactive users, and to support channel access clients that define scans differently, the first positioner can be specified by as many as six parameters: starting position (P1SP), ending position (P1EP), center position (P1CP), position width (P1WD), step increments (P1SP), and NPTS. For the other three positioners, the same parameters are available minus the NPTS field, since that applies to all. The parameters that pertain to the same positioner are a set. The record imposes an upper limit (MPTS) on NPTS. Both MPTS and NPTS are configured by the user. The positioner width, configurable in the P1WD-P4WD fields, may be negative.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NPTS	Number of Points	SHORT	Yes	100	Yes	Yes	Yes	No
MPTS	Maximum Number of Points	SHORT	Yes	100	Yes	No	No	No
P1SP	Positioner 1 Starting Point	FLOAT	Yes	0	Yes	Yes	Yes	No
P2SP	P. 2 Starting Point	FLOAT	Yes	0	Yes	Yes	Yes	No
P3SP	P. 3 Starting Point	FLOAT	Yes	0	Yes	Yes	Yes	No
P4SP	P. 4 Starting Point	FLOAT	Yes	0	Yes	Yes	Yes	No
P1EP	P. 1 Ending Point	FLOAT	Yes	0	Yes	Yes	Yes	No
P2EP	P. 2 Ending Point	FLOAT	Yes	0	Yes	Yes	Yes	No
P3EP	P. 3 Ending Point	FLOAT	Yes	0	Yes	Yes	Yes	No
P4EP	P. 4 Ending Point	FLOAT	Yes	0	Yes	Yes	Yes	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
P1CP	P. 1 Center Point	FLOAT	Yes	0	Yes	Yes	Yes	No
P2CP	P. 2 Center Point	FLOAT	Yes	0	Yes	Yes	Yes	No
P3CP	P. 3 Center Point	FLOAT	Yes	0	Yes	Yes	Yes	No
P4CP	P. 4 Center Point	FLOAT	Yes	0	Yes	Yes	Yes	No
P1WD	Positioner 1 Width	FLOAT	Yes	0	Yes	Yes	Yes	No
P2WD	Positioner 2 Width	FLOAT	Yes	0	Yes	Yes	Yes	No
P3WD	Positioner 3 Width	FLOAT	Yes	0	Yes	Yes	Yes	No
P4WD	Positioner 4 Width	FLOAT	Yes	0	Yes	Yes	Yes	No
P1SI	Positioner 1 Step Increment	FLOAT	Yes	0	Yes	Yes	Yes	No
P2SI	P. 2 Step Increment	FLOAT	Yes	0	Yes	Yes	Yes	No
P3SI	P. 3 Step Increment	FLOAT	Yes	0	Yes	Yes	Yes	No
P4SI	P. 4 Step Increment	FLOAT	Yes	0	Yes	Yes	Yes	No

Some of these fields can be redundant. For instance, the positioner width (P1WD-P4WD) is simply the distance from the starting position to the ending position (P*n*EP - P*n*SP). The record calculates redundant parameters for the same set, if the parameters are left undefined. However, the user can still configure the redundant parameters anyway.

There is no unique prescription for removing inconsistencies among redundant parameters, and no hard-coded set of preferences among parameters is likely to please everyone. Therefore, the scan record allows the user to “freeze” parameters with flags so that they will not be changed by the record's internal attempts to ensure consistency among the parameter set. Frozen parameters can be changed by the user and by any other client, but not by the record. It is the user's responsibility to ensure that frozen parameters do not prevent freely specifying unfrozen parameters. For example, if both P*n*SI and NPTS are frozen, changes to P*n*WD will be rejected. Similarly, if both P*n*SP and P*n*CP are frozen, changes to P*n*EP and PnWD will have no effect. By default, P*n*SP, P*n*SI, and NPTS are frozen. When the record cannot adjust the parameters to be consistent, a flag is raised in the alert field (ALRT) and a message reported in the state message field (SMSG).

The freeze flag override field (FFO) has two choices: `Use F-Flags` and `Override`. `Override` causes the current settings of all the freeze flags to be saved and monitors to be called for those that have changed. Use F-Flags causes the flags saved with the `Override` command to be restored if any have changed. Changing the choice of this field at run-time causes the special record support routines to perform these actions. So if `Override` is chosen at run-time, then all current settings are saved, and can be restored at a later time by changing the FFO field to `Use F-Flags`.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
FPTS	Freeze Flag for NPTS	RECCHOICE	Yes	1	Yes	Yes	No	No
FFO	Freeze Flag Override	RECCHOICE	Yes	Null	Yes	Yes	No	No
P*n*FS	Positioner n Freeze Flag for PnSP	RECCHOICE	Yes	1	Yes	Yes	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
P*n*FE	Positioner n Freeze Flag for PnEP	RECCHOICE	Yes	0	Yes	Yes	No	No
P*n*FI	Positioner n Freeze Flag for PnSI	RECCHOICE	Yes	1	Yes	Yes	No	No
P*n*FC	Positioner n Freeze Flag for PnCP	RECCHOICE	Yes	0	Yes	Yes	No	No
P*n*FW	Positioner n Freeze Flag for PnWD	RECCHOICE	Yes	0	Yes	Yes	No	No

Although this approach may seem to present the user with an overwhelming number of choices when it comes to linear scans, it should be noted that by default the user only has to configure NPTS, and the starting position (PnSP) and the step increment (PnSI) fields for each positioner in order to fully define the scan of a positioner. The operator interface (usually DM, medm or another CA client) need only present the user with these fields. However, by changing the freeze flags from the defaults and presenting the user with different fields to fill in, the scan can be defined in a completely flexible way. The result is that a simple scan can be defined easily, but advanced users are not limited in flexibility.

Lookup Mode Parameters

For those positioners using the linear mode, each desired position value (contained in the P*n*DV fields) is placed in the arrays referenced by the P*n*PA fields that are used for the lookup mode. Therefore, after the scan is complete, these arrays will always contain the sequence of positions to which the controller was commanded by the linear algorithm. For look-up mode, the user provides the values for the PnPA arrays prior to run-time. These arrays will contain no useful data if on-the-fly mode is used.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
P1PA	Positioner 1 Position Array	FLOAT[ ]	No	Null	Yes	Yes	Yes	No
P2PA	Positioner 2 Position Array	FLOAT[ ]	No	Null	Yes	Yes	Yes	No
P3PA	Positioner 3 Position Array	FLOAT[ ]	No	Null	Yes	Yes	Yes	No
P4PA	Positioner 4 Position Array	FLOAT[ ]	No	Null	Yes	Yes	Yes	No

2.3. Position Verification, Readback Process Variable, and Delta Parameters

For each positioner, the user may specify a process variable in the R1PV-R4PV fields that corresponds to the actual (or measured) position of the motor. If this readback field is configured, the scan record will confirm after each movement that the actual position is within a specified delta to the desired position. The delta is specified in the R1DL-R4DL fields. If the delta is exceeded, the scan will abort and the record will go into an alarm state. A text field within the record (SMSG) will inform the operator of the error condition.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
R1PV	Readback 1 Process Variable	STRING [40]	Yes	Null	Yes	Yes	No	No
R2PV	Readback 2 Process Variable	STRING [40]	Yes	Null	Yes	Yes	No	No
R3PV	Readback 3 Process Variable	STRING [40]	Yes	Null	Yes	Yes	No	No
R4PV	Readback 4 Process Variable	STRING [40]	Yes	Null	Yes	Yes	No	No
R1DL	Readback 1 Delta	FLOAT	Yes	0	Yes	Yes	No	No
R2DL	Readback 2 Delta	FLOAT	Yes	0	Yes	Yes	No	No
R3DL	Readback 3 Delta	FLOAT	Yes	0	Yes	Yes	No	No
R4DL	Readback 4 Delta	FLOAT	Yes	0	Yes	Yes	No	No

2.4. Detector Trigger Process Variables and Desired Command

If valid process variable names are entered into the detector trigger fields (TIPV-T2PV) fields, the scan record will write the specified desired command (a floating point number) to that process variable between the positioning phase and the data acquisition phase.

If neither detector trigger field contains a valid PV, the scan record will skip this step and acquire the data immediately. Note that specifying a *Detector Trigger* requires the scan record to be processed an additional time each point.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
T1PV	Detector Trigger 1 Process Variable	STRING [40]	Yes	Null	Yes	Yes	No	No
T2PV	Detector Trigger 2 Process Variable	STRING [40]	Yes	Null	Yes	Yes	No	No
T1CD	Trigger 1 Command	FLOAT	Yes	0	Yes	Yes	No	No
T2CD	Trigger 2 Command	FLOAT	Yes	0	Yes	Yes	No	No

3. Data Acquisition Parameters

Each scan record can acquire data from up to four process variables at each point in the scan. This data will most commonly be from a detector or from a position readback (which would record the actual motor positions at each point and could then be compared to the desired position array).

The scan results will most frequently be read as position arrays (P1PA-P4PA), which are mentioned above, and arrays of detector data (D1PV-D4PV) where each detector data element corresponds to the position element.

For single dimension scans, the scan is complete when the Execute Scan flag (the EXCS) field is set back to zero by the record processing routine (during the scan it is set to 1). The application program can then read the position arrays and the data arrays (or have a monitor set on them so the record will post its monitors when complete).

For two dimension scans similar to Figure 2, the application program should read the arrays from the SCAN_X record after the completion of each x scan and correlate it to the current y controller information. This will allow the application program to display data after each x scan. The scan record will buffer the data for only one x scan, so the application must read the arrays before the next x scan is completed. If the scan is too fast, the application program can contribute to the wait record algorithm such that the y -controllers are not moved until the application program has completely read the previous SCAN_X data.

On slow scans, the application program may want to see that the scan is processed on a point by point basis. Therefore, the scan record will post monitors on fields that it updates each point, but it will not post monitors faster than 20 times per second. If a scan is proceeding at a rate less than 20 points per second, every point will be posted. If a scan is proceeding at 100 steps per second, scalar values will be posted every 5th point (approx.). In either case, the array data will contain every point at the completion of the scan. It is not recommended that the application program use the point to point data except for keeping the operator aware of the progress of the scan.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
D1PV	Data 1 Process Variable	STRING [40]	Yes	Null	Yes	Yes	No	No
D2PV	Data 2 Process Variable	STRING [40]	Yes	Null	Yes	Yes	No	No
D3PV	Data 3 Process Variable	STRING [40]	Yes	Null	Yes	Yes	No	No
D4PV	Data 4 Process Variable	STRING [40]	Yes	Null	Yes	Yes	No	No
EXSC	Execute Scan Flag	SHORT	No	0	Yes	Yes	Yes	No
D1DA	Detector 1 Data Array	FLOAT[ ]	No	Null	Yes	No	Yes	No
D2DA	Detector 2 Data Array	FLOAT[ ]	No	Null	Yes	No	Yes	No
D3DA	Detector 3 Data Array	FLOAT[ ]	No	Null	Yes	No	Yes	No
D4DA	Detector 4 Data Array	FLOAT[ ]	No	Null	Yes	No	Yes	No

4. Operator Display Parameters

Prior to beginning an actual scan, the record can be commanded to check the scan parameters to ensure that all positioner requests are within reasonable limits. The record will do a "dry run" by calculating every positioner value (or looking it up in the table) and comparing it with the high range and low range values (P1HR-P4HR and P1LR-P4LR) associated with that positioner's Process Variable. (Drive limits are an attribute of most process variables). If any step would exceed the drive limits, the operator is notified via the SMSG field.

Other than that, the High Range and Low Range value fields are only used as the display limits for an operator interface. The same is true for the rest of these fields, which are configured to affect the information displayed to the operator. Each positioner and the detector for each positioner have the following fields:

- An Engineering Units Field (P1EU-P4EU, D1EU-D4EU), which has a string that is given to it by the user.
- A Precision Field (P1PR-P4PR, D1PR-D4PR), which holds an integer that controls the decimal precision that the corresponding field is displayed with. For instance, P1PR controls the decimal precision for positioner 1 array elements.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
P1EU	Positioner 1 Eng. Units	STRING [16]	Yes	16	Yes	Yes	No	No
P1HR	Pos. 1 High Range	FLOAT	Yes	0	Yes	Yes	No	No
P1LR	Pos. 1 Low Range	FLOAT	Yes	0	Yes	Yes	No	No
P1PR	Pos. 1 Precision	SHORT	Yes	0	Yes	Yes	No	No
P2EU	Pos. 2 Eng Units	STRING [16]	Yes	16	Yes	Yes	No	No
P2HR	Pos. 2 High Range	FLOAT	Yes	0	Yes	Yes	No	No
P2LR	Pos. 2 Low Range	FLOAT	Yes	0	Yes	Yes	No	No
P2PR	Pos. 2 Precision	SHORT	Yes	0	Yes	Yes	No	No
P3EU	Pos. 3 Eng Units	STRING [16]	Yes	16	Yes	Yes	No	No
P3HR	Pos. 3 High Range	FLOAT	Yes	0	Yes	Yes	No	No
P3LR	Pos. 3 Low Range	FLOAT	Yes	0	Yes	Yes	No	No
P3PR	Pos. 3 Precision	SHORT	Yes	0	Yes	Yes	No	No
P4EU	Pos. 4 Eng Units	STRING [16]	Yes	16	Yes	Yes	No	No
P4HR	Pos. 4 High Range	FLOAT	Yes	0	Yes	Yes	No	No
P4LR	Pos. 4 Low Range	FLOAT	Yes	0	Yes	Yes	No	No
P4PR	Pos. 4 Precision	SHORT	Yes	0	Yes	Yes	No	No
D1EU	Detector 1 Eng. Units	STRING [16]	Yes	16	Yes	Yes	No	No
D1HR	Det. 1 High Range	FLOAT	Yes	0	Yes	Yes	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
D1LR	Det. 1 Low Range	FLOAT	Yes	0	Yes	Yes	No	No
D1PR	Det. 1 Precision	SHORT	Yes	0	Yes	Yes	No	No
D2EU	Det. 2 Eng. Units	STRING [16]	Yes	16	Yes	Yes	No	No
D2HR	Det. 2 High Range	FLOAT	Yes	0	Yes	Yes	No	No
D2LR	Det. 2 Low Range	FLOAT	Yes	0	Yes	Yes	No	No
D2PR	Det. 2 Precision	SHORT	Yes	0	Yes	Yes	No	No
D3EU	Det. 3 Eng. Units	STRING [16]	Yes	16	Yes	Yes	No	No
D3HR	Det. 3 High Range	FLOAT	Yes	0	Yes	Yes	No	No
D3LR	Det. 3 Low Range	FLOAT	Yes	0	Yes	Yes	No	No
D3PR	Det. 3 Precision	SHORT	Yes	0	Yes	Yes	No	No
D4EU	Det. 4 Eng. Units	STRING [16]	Yes	16	Yes	Yes	No	No
D4HR	Det. 4 High Range	FLOAT	Yes	0	Yes	Yes	No	No
D4LR	Det. 4 Low Range	FLOAT	Yes	0	Yes	Yes	No	No
D4PR	Det. 4 Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

5. Run-time Parameters

These fields are used to process the record, to implement monitors for certain fields, and/or to keep track of data for processing and/or for the operator. None of these fields are configurable by a database configuration tool. Most of them can be accessed at run-time, and many can be modified at run-time.

The Code Version (VERS) field reflects the version of scan record processing routines.

The VAL field is not used.

The State Message (SMSG) field holds a message sent by the record that alerts the operator to an error condition. It can be cleared by writing a 0 to the Command (CMND) field. The CMND field sends two commands: Clear the State Message field (SMSG), which corresponds to 0; and Execute a “dry run”, checking the desired position against the range limits for each positioner, this command corresponding to 1.

The Alert (ALRT) field is a flag which indicates if an error condition currently exists. 1 means YES; 0, NO. The cause of the condition will be displayed in the SMSG field.

The Current Point (CPT) field contains the current point of an active scan. The Desired Value fields for each positioner (P1DV-P4DV) contain the desired value of each positioner for the current point (CPT) in the scan. The Readback Current Value (R1CV-R4CV) fields contain the current readback value for each positioner. The Detector Current Value (D1CV-D4CV) contain each detector's current value for the current point in the scan. The event posting for these fields is throttled to 20 Hz, so for fast scans not every value will be posted.

The PCPT, PXSC, P1LV-P4LV, R1LV-R4LV, and D1LV-D4LV fields all contain the previous or "last" value for their corresponding fields. For instance, the R1LV field contains the last value for the R1CV field. These fields are used to implement monitors for the corresponding field. For instance, if CPT does not equal PCPT when the record is processed, then monitors are triggered for CPT.

The Name Valid fields (*xx*NV) are flag fields which indicate if the corresponding process variable field contains an existing process variable. For instance, the P1NV field indicates if the P1PV field contains valid process variable; the R4NV field indicates whether R4PV contains a valid, existing process variable, and so on. These fields are used to decide if a process variable has been configured by the user. If not, the fields associated with that variable are ignored.

The Database Address fields (*xx*DB) are of normally of interest only to the record itself, and are not even accessible at run-time. They contain pointers to the dbAddr structures of the corresponding process variables. For instance, P1DB points to the dbAddr structure of P1PV.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VERS	Code Version	FLOAT	No	1.0	Yes	No	No	No
VAL	Value Field	DOUBLE	No	0	Yes	Yes	No	No
SMSG	State Message	STRING [40]	No	Null	Yes	Yes	Yes	No
CMND	Command Field	ENUM	No	0	Yes	Yes	Yes	No
ALRT	Alert Field	UCHAR	No	0	Yes	No	Yes	No
RPVT	Record Private	NOACCESS	No	Null	No	No	No	No
PXSC	Previous Execute Scan	UCHAR	No	0	Yes	No	No	No
CPT	Current Point	SHORT	No	0	Yes	No	Yes*	No
PCPT	Previous Current Point	SHORT	No	0	Yes	No	No	No
TOLP	Time of Last Posting	ULONG	No	0	Yes	No	No	No
P1NV	Pos. 1 Name Valid	LONG	No	0	Yes	Yes	Yes	No
P2NV	Pos. 2 Name Valid	LONG	No	0	Yes	Yes	Yes	No
P3NV	Pos. 3 Name Valid	LONG	No	0	Yes	Yes	Yes	No
P4NV	Pos. 4 Name Valid	LONG	No	0	Yes	Yes	Yes	No
R1NV	Readback 1 Name Valid	LONG	No	0	Yes	Yes	Yes	No
R2NV	Rbk. 2 Name Valid	LONG	No	0	Yes	Yes	Yes	No
R3NV	Rbk. 3 Name Valid	LONG	No	0	Yes	Yes	Yes	No
R4NV	Rbk. 4 Name Valid	LONG	No	0	Yes	Yes	Yes	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
T1NV	Trigger 1 Name Valid	LONG	No	0	Yes	Yes	Yes	No
T2NV	Trigger 2 Name Valid	LONG	No	0	Yes	Yes	Yes	No
D1NV	Data 1 Name Valid	LONG	No	0	Yes	Yes	Yes	No
D2NV	Data 2 Name Valid	LONG	No	0	Yes	Yes	Yes	No
D3NV	Data 3 Name Valid	LONG	No	0	Yes	Yes	Yes	No
D4NV	Data 4 Name Valid	LONG	No	0	Yes	Yes	Yes	No
P1DV	Pos. 1 Desired Value	FLOAT	No	0	Yes	No	Yes*	No
P1LV	Pos. 1 Last Value	FLOAT	No	0	Yes	No	No	No
R1CV	Readback 1 Current Value	FLOAT	No	0	Yes	No	Yes*	No
R1LV	Readback 1 Last Value	FLOAT	No	0	Yes	No	No	No
P2DV	Pos. 2 Desired Value	FLOAT	No	0	Yes	No	Yes*	No
P2LV	Pos. 2 Last Value	FLOAT	No	0	Yes	No	No	No
R2CV	Readback 4 Current Value	FLOAT	No	0	Yes	No	Yes*	No
R2LV	Readback 2 Last Value	FLOAT	No	0	Yes	No	No	No
P3DV	Pos. 3 Desired Value	FLOAT	No	0	Yes	No	Yes*	No
P3LV	Pos. 3 Last Value	FLOAT	No	0	Yes	No	No	No
R3CV	Readback 4 Current Value	FLOAT	No	0	Yes	No	Yes*	No
R3LV	Readback 3 Last Value	FLOAT	No	0	Yes	No	No	No
P4DV	Pos. 4 Desired Value	FLOAT	No	0	Yes	No	Yes*	No
P4LV	Pos. 4 Last Value	FLOAT	No	0	Yes	No	No	No
R4CV	Readback 4 Current Value	FLOAT	No	0	Yes	No	Yes*	No
R4LV	Readback 4 Last Value	FLOAT	No	0	Yes	No	No	No
D1CV	Detector 1 Current Value	FLOAT	No	0	Yes	No	Yes*	No
D1LV	Detector 1 Last Value	FLOAT	No	0	Yes	No	No	No
D2CV	Detector 2 Current Value	FLOAT	No	0	Yes	No	Yes*	No
D2LV	Detector 2 Last Value	FLOAT	No	0	Yes	No	No	No
D3CV	Detector 3 Current Value	FLOAT	No	0	Yes	No	Yes*	No
D3LV	Detector 3 Last Value	FLOAT	No	0	Yes	No	No	No
D4CV	Detector 4 Current Value	FLOAT	No	0	Yes	No	Yes*	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
D4LV	Detector 4 Last Value	FLOAT	No	0	Yes	No	No	No
P1DB	Pos. 1 dbAddr	NOACCESS	No	Null	No	No	No	No
P2DB	Pos. 2 dbAddr	NOACCESS	No	Null	No	No	No	No
P3DB	Pos. 3 dbAddr	NOACCESS	No	Null	No	No	No	No
P4DB	Pos. 4 dbAddr	NOACCESS	No	Null	No	No	No	No
R1DB	Readback 1 dbAddr	NOACCESS	No	Null	No	No	No	No
R2DB	Readback 2 dbAddr	NOACCESS	No	Null	No	No	No	No
R3DB	Readback 3 dbAddr	NOACCESS	No	Null	No	No	No	No
R4DB	Readback 4 dbAddr	NOACCESS	No	Null	No	No	No	No
T1DB	Trigger 1 dbAddr	NOACCESS	No	Null	No	No	No	No
T2DB	Trigger 2 dbAddr	NOACCESS	No	Null	No	No	No	No
D1DB	Detector 1 dbAddr	NOACCESS	No	Null	No	No	No	No
D2DB	Detector 2 dbAddr	NOACCESS	No	Null	No	No	No	No
D3DB	Detector 3 dbAddr	NOACCESS	No	Null	No	No	No	No
D4DB	Detector 4 dbAddr	NOACCESS	No	Null	No	No	No	No

Chapter 29:sel - Select

1. Introduction

The select record computes a value based on input obtained from up to 12 locations. The selection algorithm can be one of the following: `Specified`, `High Signal`, `Low Signal`, `Median Signal`. Each input can be a constant, a database link, or a channel access link.

The fields in this record fall into several categories:

scan parameters

read parameters

select parameters

operator display parameters

alarm parameters

monitor parameters

run-time parameters

2. Scan Parameters

The select record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields work.

3. Read Parameters

The INPA-L links determine where the selection record retrieves the values from which it is to select or compute its final value. The INPA-L links are input links configured by the user to be either constants, channel access links, or database links. If channel access or database links, a value is retrieved for each link and placed in the corresponding value field, A-L. If any input link is a constant, the value field for that link will be initialized with the constant value given to it and can be modified via dbPuts. See *Address Specification, Chapter 1, 2*, for information on how to specify database links.

Any links not defined are ignored by the selection record and its algorithm. An undefined link is any constant link whose value is 0. At initialization time, the corresponding value links for such fields are set equal to le30, which means MISSING. The value field of an undefined link can be changed at run-time from the MISSING value to another value in order to define the link and its field. Note that all undefined links must be recognized as such if the selection algorithm is to work as expected.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
INPA	Input A	INLINK	Yes	0	No	No	N/A	No
INPB	Input B	INLINK	Yes	0	No	No	N/A	No
INPC	Input C	INLINK	Yes	0	No	No	N/A	No
INPD	Input D	INLINK	Yes	0	No	No	N/A	No
INPE	Input E	INLINK	Yes	0	No	No	N/A	No
INPF	Input F	INLINK	Yes	0	No	No	N/A	No
INPG	Input G	INLINK	Yes	0	No	No	N/A	No
INPH	Input H	INLINK	Yes	0	No	No	N/A	No
INPI	Input I	INLINK	Yes	0	No	No	N/A	No
INPJ	Input J	INLINK	Yes	0	No	No	N/A	No
INPK	Input K	INLINK	Yes	0	No	No	N/A	No
INPL	Input L	INLINK	Yes	0	No	No	N/A	No
A	Input A Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
B	Input B Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
C	Input C Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
D	Input D Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
E	Input E Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
F	Input F Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
G	Input G Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
H	Input H Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
I	Input I Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
J	Input J Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
K	Input K Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
L	Input L Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes

4. Select Parameters

The selection algorithm is determined by three fields configurable by the user: the select mechanism (SELM) field, the select number (SELN) field, and the index value location (NVL) field.

The SELM field has four choices, i.e., four algorithms: `Specified`, `High Signal`, `Low Signal`, and `Median Signal`. The selection record's VAL field is determined differently for each algorithm. For `Specified`, the VAL field is set equal to the value field (A, B, C, D, E, F, G, H, I, J, K, or L) specified by the SELN field. The SELN field contains a number from 0-11 which corresponds to the value field to be used (0 means use A; 1 means use B, etc.). How the NVL field is configured determines, in turn, SELN's value. NVL is an input link from which a value for SELN can be retrieved, Like most other input links NVL can be a constant, or a channel access or database link. If NVL is a link, SELN is retrieved from the location in NVL. If a constant, SELN is initialized to the value given to the constant and can be changed via dbPuts. See *Address Specification, Chapter 1, 2*, for information on how to specify database links.

The `High Signal`, `Low Signal`, and `Median Signal` algorithms do not use SELN or NVL. If `High Signal` is chosen, VAL is set equal to the highest value out of all the defined value fields (A-F). If `Low Signal` is chosen, VAL is set equal to lowest value of all the defined fields (A-F). And if `Median Signal` is chosen, VAL is set equal to the median value of the defined value fields (A-F). (Note that these algorithms select from the value fields; they do not select from the value field index. For instance, `Low Signal` will not select the A field's value unless the value itself is the lowest of all the defined values.)

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SELM	Select Mechanism	RECCHOICE	Yes	0	Yes	Yes	No	No
SELN	Select Number	USHORT	No	0	Yes	Yes	No	No
NVL	Index Value Location, an input link	INLINK	Yes	0	No	No	N/A	No

5. Operator Display Parameters

These parameters are used to present meaningful data to the operator. They display the value and other parameters of the select record either textually or graphically.

EGU is a string of up to 16 characters describing the units that the selection record manipulates. It is retrieved by the `get_units` record support routine.

The HOPR and LOPR fields set the upper and lower display limits for the VAL, HIHI, HIGH, LOW, and LOLO fields. Both the `get_graphic_double` and `get_control_double` record support routines retrieve these fields.

The PREC field determines the floating point precision with which to display VAL. It is used whenever the `get_precision` record support routine is called.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	null	Yes	Yes	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

The possible alarm conditions for select records are the SCAN, READ, and limit alarms. The SCAN and READ alarms are called by the record or device support routines. The limit alarms are configured by the user in the HIHI, LOLO, HIGH, and LOW fields using numerical values. They specify conditions for the VAL field. For each of these fields, there is a corresponding severity field which can be either NO_ALARM, MINOR, or MAJOR. See *Alarm Specification, Chapter 1, 4*, for a complete explanation of alarms and these fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modif y	Rec Proc Monitor	PP
HIHI	Hihi Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HIGH	High Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOW	Low Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOLO	Lolo Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HHSV	Hihi Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HSV	High Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LSV	Low Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LLSV	Lolo Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HYST	Alarm Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

7. Monitor Parameters

These fields are configurable by the user. They are used as deadbands for the archiver and monitor calls for the VAL field. Unless, VAL changes by more than the value specified by each, then the respective monitors will not be called. If these fields have a value of zero, everytime the VAL changes, monitors are triggered; if they have a value of -1, everytime the record is processed, monitors are triggered. *Monitor Specification, Chapter 1, 5*, gives a complete explanation of alarms and deadbands.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ADEL	Archive Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
MDEL	Monitor, i.e. value change, Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

8. Run-time Parameters

These parameters are used by the run-time code for processing the selection record. They are not configurable prior to run-time, nor are they modifiable at run-time. They represent the current state of the record. The record support routines use some of them for more efficient processing.

The VAL field is the result of the selection record's processing. It can be accessed in the normal way by another record or through database access, but is not modifiable except by the record itself. The LALM, ALST, and the MLST are used to implement the HYST, ADEL, and MDEL hysteresis factors for the alarms, archiver, and monitors, respectively.

The LA-LL fields are used to implement the monitors for each of the value fields, A-F. They represent previous input values. For example, unless LA is not equal to A, no monitor is invoked for A.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Value Field	DOUBLE	No	0	Yes	No	Yes	No
LALM	Last Alarmed Value	DOUBLE	No	0	Yes	No	No	No
ALST	Archive Last Value	DOUBLE	No	0	Yes	No	No	No
MLST	Monitor Last Value	DOUBLE	No	0	Yes	No	No	No
LA	Last A Value	DOUBLE	No	0	Yes	No	No	No
LB	Last B Value	DOUBLE	No	0	Yes	No	No	No
LC	Last C Value	DOUBLE	No	0	Yes	No	No	No
LD	Last D Value	DOUBLE	No	0	Yes	No	No	No
LE	Last E Value	DOUBLE	No	0	Yes	No	No	No
LF	Last F Value	DOUBLE	No	0	Yes	No	No	No
LG	Last G Value	DOUBLE	No	0	Yes	No	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LH	Last H Value	DOUBLE	No	0	Yes	No	No	No
LI	Last I Value	DOUBLE	No	0	Yes	No	No	No
LJ	Last J Value	DOUBLE	No	0	Yes	No	No	No
LK	Last K Value	DOUBLE	No	0	Yes	No	No	No
LL	Last L Value	DOUBLE	No	0	Yes	No	No	No

9. Record Support Routines

init_record

IF NVL is a constant, SELN is set to its value. If NVL is a PV_LINK a channel access link is created.

For each constant input link, the corresponding value field is initialized with the constant value (or 1e30 if the constant has the value 0).

For each input link that is of type PV_LINK, a channel access link is created.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_units

Retrieves EGU.

get_precision

Retrieves PREC.

get_graphic_double

Sets the upper display and lower display limits for a field. If the field is VAL, HIHI, HIGH, LOW, or LOLO, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_control_double

Sets the upper control and the lower control limits for a field. If the field is VAL, HIHI, HIGH, LOW, or LOLO, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_alarm_double

Sets the following values:

upper_alarm_limit = HIHI

upper_warning_limit = HIGH

lower_warning_limit = LOW

lower_alarm_limit = LOLO

10. Record Processing

Routine process implements the following algorithm:

1. If NVL is a database or channel access link, SELN is obtained from NVL. Fetch all values if database or channel access links. If SELM is SELECTED, then only the selected link is fetched.
2. Implement the appropriate selection algorithm. For SELECT_HIGH, SELECT_LOW, and SELECT_MEDIAN, input fields are ignored if they are undefined. If success, UDF is set to FALSE.
3. Check alarms. This routine checks to see if the new VAL causes the alarm status and severity to change. If so, NSEV, NSTA, and LALM are set. It also honors the alarm hysteresis factor (HYST). Thus the value must change by more than HYST before the alarm status and severity is lowered.
4. Check to see if monitors should be invoked.
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - Archive and value change monitors are invoked if ADEL and MDEL conditions are met
    - Monitors for A-L are checked whenever other monitors are invoked
    - NSEV and NSTA are reset to 0.
5. Scan forward link if necessary, set PACT FALSE, and return.

Chapter 30:seq - Sequence

1. Introduction

The Sequence record is used to trigger the processing of up to ten other records and send values to those records. It is similar to the fanout record, except that it will fetch an input value and write an output value instead of simply processing a collection of forward links. It can also specify one of several selection algorithms that determine which values to write. It has no associated device support. Its fields fall into the following categories:

scan parameters

desired output parameters

output parameters

selection algorithm parameters

delay parameters

operator display parameters

run-time parameters

2. Scan Parameters

The sequence record has the standard fields for specifying under what circumstances it will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used.

3. Desired Output Parameters

These fields determine where the record retrieves the values it is to write to other records. All of these values are not necessarily used, depending on the selection algorithm.

The sequence record can retrieve up to 10 values from 10 locations. The user specifies the locations in the Desired Output Link fields (DOL1-DOLA), which can be either constants, database links, or channel access links. If a Desired Output Link is a constant, the corresponding value field for that link is initialized to the constant value and *cannot* be changed via dbputs. Otherwise, if the Desired Output Link is a database or channel access link, a value is fetched from the link each time the record is processed (provided that the output link is part of the record's selection algorithm). See *Address Specification, Chapter 1, 2*, for information on how to specify database links.

The value fetched from the Desired Output Links are stored in the corresponding Desired Output Value fields (DO1-DOA). These fields can be initialized to a constant value, but they cannot be changed via dbPuts.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
DOL1	Desired Output Link 1	INLINK	Yes	0	No	No	N/A	No
DOL2	Desired Output Link 2	INLINK	Yes	0	No	No	N/A	No
DOL3	Desired Output Link 3	INLINK	Yes	0	No	No	N/A	No
DOL4	Desired Output Link 4	INLINK	Yes	0	No	No	N/A	No
DOL5	Desired Output Link 5	INLINK	Yes	0	No	No	N/A	No
DOL6	Desired Output Link 6	INLINK	Yes	0	No	No	N/A	No
DOL7	Desired Output Link 71	INLINK	Yes	0	No	No	N/A	No
DOL8	Desired Output Link 8	INLINK	Yes	0	No	No	N/A	No
DOL9	Desired Output Link 9	INLINK	Yes	0	No	No	N/A	No
DOLA	Desired Output Link 10	INLINK	Yes	0	No	No	N/A	No
DO1	Desired Output Value, Link 1	DOUBLE	No	0	Yes	Yes	No	No
DO2	Desired Output Value, Link 2	DOUBLE	No	0	Yes	Yes	No	No
DO3	Desired Output Value, Link 3	DOUBLE	No	0	Yes	Yes	No	No
DO4	Desired Output Value, Link 4	DOUBLE	No	0	Yes	Yes	No	No
DO5	Desired Output Value, Link 5	DOUBLE	No	0	Yes	Yes	No	No
DO6	Desired Output Value, Link 6	DOUBLE	No	0	Yes	Yes	No	No
DO7	Desired Output Value, Link 7	DOUBLE	No	0	Yes	Yes	No	No
DO8	Desired Output Value, Link 8	DOUBLE	No	0	Yes	Yes	No	No
DO9	Desired Output Value, Link 9	DOUBLE	No	0	Yes	Yes	No	No
DOA	Desired Output Value, Link 10	DOUBLE	No	0	Yes	Yes	No	No

4. Output Parameters

When the record is processed, the desired output values are retrieved for the links in the record's selection algorithm and are written to the corresponding output link (LNK1-LNKA). These output links can be database links or channel access links; they cannot be device addresses. There are ten output links, one for each desired output link. Only those that are defined are used.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LNK1	Output link 1	OUTLINK	Yes	0	No	No	N/A	No
LNK2	Output link 2	OUTLINK	Yes	0	No	No	N/A	No
LNK3	Output link 3	OUTLINK	Yes	0	No	No	N/A	No
LNK4	Output link 4	OUTLINK	Yes	0	No	No	N/A	No
LNK5	Output link 5	OUTLINK	Yes	0	No	No	N/A	No
LNK6	Output link 6	OUTLINK	Yes	0	No	No	N/A	No
LNK7	Output link 7	OUTLINK	Yes	0	No	No	N/A	No
LNK8	Output link 8	OUTLINK	Yes	0	No	No	N/A	No
LNK9	Output link 9	OUTLINK	Yes	0	No	No	N/A	No
LNKA	Output link 1	OUTLINK	Yes	0	No	No	N/A	No

5. Selection Algorithm Parameters

When the sequence record is processed, it uses a selection algorithm similar to that of the selection record to decide which links to process.The select mechanism field (SELM) has four algorithms to choose from: `All`, `Specified` or `Mask`.

The `All` algorithm causes the record to process each input and output link each time the record is processed, in order from 1 to 10. So when SELM is `All`, the desired output value from DOL1 will fetched and sent to LNK1, then the desired output value from DOL2 will be fetched and sent to the location in LNK2, and so on until the last input and output link DOA and LNKA. (Note that undefined links are not used.) If DOL*x* is a constant, the current value field is simply used and the desired output link is ignored. The SELN field is not used when `All` is the algorithm.

When the `Specified` algorithm is chosen, each time the record is processed it gets the integer value in the Link Selection (SELN) field and uses that as the index of the link to process. For instance, if SELN is 4, the desired output value from DO4 will be retrieved and sent to LNK4. If DOL*x* is a constant, DO*x* is simply used without the value being fetched from the input link.

When Mask is chosen, each time the record is processed, the record uses the integer value from the SELN field as a mask to determine the links to process. For instance, if SELN is 1, then the value from DO1 will be written to the location in LNK1. If SELN is 3, the record will retrieve the values from DO1 and DO2 and write them to the locations in LNK1 and LNK2, respectively. If SELN is 63, DO1...DO6 will be written to LNK1...LNK6.

LNK2. If SELN is 63, DO1...DO6 will be written to LNK1...LNK6.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SELM	Select Mechanism	RECCHOICE	Yes	0	Yes	Yes	No	No
SELN	Link Selection	USHORT	No	1	Yes	Yes	No	No
SELL	Link Selection Location	INLINK	Yes	0	No	No	N/A	No

6. Delay Parameters

The delay parameters consist of 10 fields, one for each I/O link discussed above. These fields can be configured to cause the record to delay processing the link. For instance, if the user gives the DLY1 field a value of 3.0, each time the record is processed at run-time, the record will delay processing the DOL1, DOV1, and LNK1 fields for three seconds. That is, the desired output value will not be fetched and written to the output link until three seconds have lapsed.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
DLY1	Delay time	DOUBLE	Yes	0	Yes	Yes	No	No
DLY2	Delay time	DOUBLE	Yes	0	Yes	Yes	No	No
DLY3	Delay time	DOUBLE	Yes	0	Yes	Yes	No	No
DLY4	Delay time	DOUBLE	Yes	0	Yes	Yes	No	No
DLY5	Delay time	DOUBLE	Yes	0	Yes	Yes	No	No
DLY6	Delay time	DOUBLE	Yes	0	Yes	Yes	No	No
DLY7	Delay time	DOUBLE	Yes	0	Yes	Yes	No	No
DLY8	Delay time	DOUBLE	Yes	0	Yes	Yes	No	No
DLY9	Delay time	DOUBLE	Yes	0	Yes	Yes	No	No
DLYA	Delay time	DOUBLE	Yes	0	Yes	Yes	No	No

7. Operator Display Parameters

These parameters are used to present meaningful data to the operator. The Precision field (PREC) determines the decimal precision for the VAL field when it is displayed. It is used when the `get_precision` record routine is called.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

8. Alarm Parameters

The sequence record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

9. Record Support Routines

The only record support routine is process.

1. First, PACT is set to TRUE, and the link selection is fetched. Depending on the selection mechanism, the link selection output links are processed in order from LNK1 to LNKA. When LNK*n* is processed, the corresponding DLY*n* value is used to generate a delay via watchdog timer.
2. After DLY*n* seconds have expired, the input value is fetched from DO*n* (if DOL*n* is constant) or DOL*n* (if DOL*n* is a database link or channel access link) and written to LNK*n*.
3. When all links are completed, an asynchronous completion call back to dbProcess is made (see the Application Developer's Guide for more information on asynchronous processing.)
4. Then UDF is set to FALSE.
5. Monitors are checked.
6. The forward link is scanned, PACT is set FALSE, and the process routine returns.

For the delay mechanism to operate properly, the record is processed asynchronously. The only time the record will not be processed asynchronously is when there are no non-NULL output links selected (i.e. when it has nothing to do.) The processing of the links is done via callback tasks at the priority set in the PRIO field in dbCommon (see the Application Developer's Guide for more information on call

Chapter 31:State

1. Introduction

The state record is a means for a state program to communicate with the operator interface. Its only function is to provide a place in the database through which the state program can inform the operator interface of its state by storing an arbitrary ASCII string in its VAL field. The state record fields fall into the following categories:

scan parameters

operator display parameters

other parameters

2. Scan Parameters

The state record has the standard fields for specifying under what circumstances it will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used.

3. Operator Display Parameters

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

4. Alarm Parameters

The state record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

5. Run-time Parameters

These parameters are used by the application code to convey the state of the program to the operator interface. The VAL field holds the string retrieved from the state program. The OVAL is used to implement monitors for the VAL field. When the string in OVAL differs from the one in VAL, monitors are triggered. They represent the current state of the sequence program.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Value Field	STRING [20]	Yes	Null	Yes	Yes	Yes	Yes
OVAL	Old Value	STRING [20]	No	Null	Yes	No		

6. Record Support Routines

Two record support routines are provided:

process

process triggers monitors on VAL when it changes and scans the forward link if necessary.

get_value

get_value fills in struct valueDes so that it refers to VAL.

Chapter 32: Stepper Motor

1. Introduction

The stepper motor record controls and monitors stepper motors. The stepper motor can be controlled in either *position* mode or *velocity* mode. Position mode moves a device to a position. Velocity mode causes the motor to revolve continuously.

The record is set up assuming that the same stepper motor will not be used in both velocity and position mode. The fields for the stepper motor record fall into the following categories:

scan parameters

setup parameters

desired output parameters

output and readback parameters

operator display parameters

alarm parameters

monitor parameters

run-time parameters.

2. Scan Parameters

The stepper motor record has the standard fields for specifying under what circumstances the record will be processed. A motor record that is scanned periodically or on event will compare its actual position to the desired position and will attempt to achieve its position. It will only make a certain number of attempts; the number of attempts is specified in the retry (RTRY) field. A passive motor will not go through the retry logic.

These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Setup Parameters

The set-up parameters determine how the stepper motor is initialized and how it operates at run-time.

The MODE field has two choices: `Velocity` or `Position`.

The VELO field determines the velocity that the stepper motor will run at when in the velocity mode. In position mode, it determines the pulses per second to move when it moves to a new position.

The ACCL field determines the acceleration of the motor by specifying a number of seconds. In velocity mode, the stepper motor will accelerate at the rate needed to reach the operating velocity in the number of seconds specified in ACCL. In positional mode, ACCL determines the seconds the stepper motor will take to reach the specified velocity, and the number of seconds it will take to come to a stop at the new position.

The MRES field specifies the number of pulses per revolution for the stepper motor, while ERES specifies the encoder pulses per revolution.

The DIST field's value determines the distance which the stepper motor moves for each pulse in engineering units.

The RTRY field specifies the number of attempts the record will make to reach the desired position each time it's processed, and the RDBD field specifies a deadband for this processing. Thus, when the motor is moving in a positive direction, no retries will be attempted unless the readback value is greater than VAL added to the value in the RDBD field. If the motor is moving in the negative direction, no retries will be attempted unless the readback value is less than VAL added to the value in the RDBD field:

```
if direction positive and RBV > (VAL + RDBD)
      attempt to correct position RTRY times
if direction negative and RBV < (VAL + RDBD)
      attempt to correct position RTRY times
```

The initialization algorithm (IALG) field has three algorithms to choose from: `No Initialization`, `Move to the Positive Limit`, and `Move to the Negative Limit`. The IVAL field determines the value of VAL at initialization; i.e., VAL is initialized to the value in IVAL when IALG is either Move to Positive or Move to Negative.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ACCL	Seconds to Reach Velocity	FLOAT	Yes	0	Yes	Yes	No	No
VELO	Velocity in Pulses per Second	FLOAT	Yes	0	Yes	Yes	No	No
DIST	Distance in Engineering Units of 1 Pulse	FLOAT	Yes	0	Yes	Yes	No	No
IVAL	Initial Value	FLOAT	Yes	0	Yes	Yes	No	No
MODE	Mode—Position/Velocity	RECCHOICE	Yes	0	Yes	Yes	No	No
IALG	Initialization Algorithm	RECCHOICE	Yes	0	Yes	Yes	No	No
RTRY	Number Of Retries Before Failure	SHORT	Yes	0	Yes	Yes	No	No
RDBD	Retry Deadband	FLOAT	Yes	0	Yes	Yes	No	No
MRES	Motor Pulses per Revolution	USHORT	Yes	0	Yes	Yes	No	No
ERES	Encoder Pulses per Revolution	USHORT	Yes	0	Yes	Yes	No	No

4. Desired Output Parameters

The stepper motor record specifies where the desired output originates in the DOL field. The desired output needs to be in engineering units. If the output is controlled by the operator or a sequence program, nothing should be entered in the DOL field.

The desired output link (DOL) and operation mode select (OMSL) fields work as they do for output records. If OMSL is `closed_loop`, the desired output (VAL) is fetched from the DOL link, which can be a database or channel access link. If `supervisory`, VAL can be changed via dbPuts. DOL can be a constant, in which case VAL is initialized to the constant. If DOL is a constant, OMSL cannot specify `closed_loop`. See *Address Specification, Chapter 1, 2*, for information on specifying links.

VAL indicates the position of the stepper motor in position mode; in velocity mode, it indicates the current velocity. When the desired output is retrieved and placed in VAL, it is forced to be within the drive limits configured in the DRVH and DRVL fields. The DRVH and DRVL fields must be given values by the user. **Note: If nothing is entered in the DRVH and DRVL fields, the motor will not move.** The `get_control_double()` record support routine returns the values in these fields when it is called.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Value	FLOAT	No	0	Yes	Yes	Yes	Yes
DOL	Desired Output Location (Input Link)	INLINK	Yes	0	No	No	N/A	No
OMSL	Output Mode Select	GBLCHOICE	Yes	0	Yes	Yes	No	No

5. Output and Readback Parameters

In the OUT output link field, the user must enter the address of the I/O card of the device, and in the DTYP field, the user must enter the name of the appropriate stepper motor device support module. Be aware that the address format differs according to the I/O bus used. See *Address Specification, Chapter 1, 2*, for information on hardware addresses. The user can see a list of the device support modules currently supported at the user's local site by using the `dbst` utility in R3.13.

The RDBL field can contain the address of an encoder or an external device, such as an LDVT, or it can specify the motor position field (MPOS) or the encoder position field (EPOS) in the same record. In the case of an external readback, the value would be read through an analog input record and linked to this record through this field. The value retrieved from this field is placed into the RBV field.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OUT	Output Link	OUTLINK	Yes	0	No	No	N/A	No
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	
RDBL	Read Back Location (an input link)	INLINK	Yes	0	No	No	N/A	No
RBV	Read Back Value	FLOAT	No	0	Yes	Yes	Yes	No

6. Operator Display Parameters

These parameters are used to present meaningful data to the operator. These fields are used to display the value and other parameters of the stepper motor either textually or graphically.

EGU is a string of up to 16 characters describing the engineering units of VAL. It is retrieved by the `get_units` record support routine.

The HOPR and LOPR fields set the upper and lower display limits for the VAL, HIHI, HIGH, LOW, LOLO, MPOS, EPOS, RBV, and LVAL fields. Only the `get_graphic_double` record support routine retrieves these fields. If these values are defined, they must be in the range: DRVL<=LOPR<=HOPR<=DRVH.

The PREC field determines the floating point precision with which to display the VAL and LVAL fields. It is used whenever the `get_precision` record support routine is called.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	null	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

7. Alarm Parameters

The possible alarm conditions for stepper motor records are the SCAN, READ, and limit alarms. The SCAN and READ alarms are called by the record or device support routines and are always of MAJOR severity.

The limit alarms are configured by the user in the HIHI, LOLO, HIGH, and LOW fields using numerical values. The conditions apply to the VAL and RBV fields of the stepper motor record. For each of these fields, there is a corresponding severity field which can specify either NO ALARM, MINOR, or MAJOR. The user can specify a deadband around these limits in the HYST or hysteresis field. See *Alarm Specification, Chapter 1, 4*, for a complete explanation of analog alarms and these fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HIHI	Hihi Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HIGH	High Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOW	Low Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOLO	Lolo Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HHSV	Hihi Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HSV	High Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LSV	Low Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LLSV	Lolo Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HYST	Alarm Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

8. Monitor Parameters

These parameters are used to determine when to send monitors placed on the VAL field. The monitors are sent when the value field exceeds the last monitored field by the appropriate delta. If these fields have a value of zero, everytime the value changes, a monitor will be triggered; if they have a value of -1, everytime the record is processed, monitors are triggered. The ADEL field is used by archive monitors and the MDEL field for all other types of monitors. See *Monitor Specification, Chapter 1, 5*, for a complete explanation of monitors.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ADEL	Archive Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
MDEL	Monitor, i.e. value change, Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

9. Run-time Parameters

These fields are not configurable by the user before run-time, but most of them can be modified after initialization.

The STHM field will set the current position of the motor to its home or zero position when this field is set to 1 via a dbPut. Its value is automatically reset to 0 after the command is accepted.

Setting the STOP field to 1 will cause the motor to stop moving. Its value is also reset to 0 after the command is accepted.

The DMOV field is a flag which indicates if the motor is done moving and all retries are complete. It is significant only while the record is in POSITION mode. It will be 1 if TRUE and 0 if FALSE.

The RVAL field contains the value which is actually written to output after it is forced in the drive limits.

The RBV and RRBV fields contain the value retrieved from the readback location link (RDBL).

The MOVN field indicates whether the motor is currently moving. It will contain a 1 if moving or a 0 if not moving. The DIR field indicates which direction the record is moving—a 0 means clockwise and a 1 means counter clockwise.

The CW and CCW fields contain the values for the clockwise and counterclockwise limit switches, respectively. The MCW and MCCW fields will be 1 if the limit has been reached or 0 if not.

The CVEL field indicates whether the motor is at constant velocity. If so, the value of this field is 1.

The RCNT field will indicate the current number of retries that have been attempted.

The POSM field will contain a 1 if the motor is moving in a positive direction and a 0 if otherwise.

The MISS field holds the error after the first attempt has been made to change the stepper motor's position, the error being the difference between VAL and RBV, the readback value. If monitors are present, the value is posted.

The EPOS field holds the value obtained from encoder readback. The MPOS holds the readback value of the motor positions.

The MLST and ALST fields are used to implement monitors on the VAL field. If the difference between one of these fields and the VAL field exceeds the appropriate deadband (MDEL, for instance) monitors are triggered.

The LVEL, LACC, and LVAL fields contain the last values set for the VELO, ACCL, and VAL fields respectively. They are used to determine when the velocity and acceleration should be recalculated.

The CMOD or current mode fields indicates which mode, position or velocity, the record is using.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
STHM	Set Home	SHORT	No	0	Yes	Yes	No	Yes
STOP	Stop	SHORT	No	0	Yes	Yes	No	Yes
DMOV	Done Moving to Value	SHORT	No	0	Yes	Yes	No	No
RVAL	Raw Data Value	LONG	No	0	Yes	Yes	Yes	Yes
RRBV	Raw Read Back Value	LONG	No	0	Yes	Yes	Yes	No
INIT	Initialize	SHORT	No	0	Yes	Yes	No	Yes
MOVN	Moving Status	SHORT	No	0	Yes	Yes	No	No
DIR	Current Direction	SHORT	No	0	Yes	Yes	No	Yes
MCW	Motor Clockwise Limit Switch Value	SHORT	No	0	Yes	Yes	No	Yes
MCCW	Motor Counter Clockwise Limit Switch Value	SHORT	No	0	Yes	Yes	No	Yes
CW	Clockwise Limit	SHORT	No	0	Yes	Yes	No	Yes
CCW	Counter Clockwise Limit	SHORT	No	0	Yes	Yes	No	Yes
CVEL	Constant Velocity	SHORT	No	0	Yes	Yes	No	No
RCNT	Current Retry Count	SHORT	No	0	Yes	Yes	Yes	No
POSM	Positive Motion	FLOAT	No	0	Yes	No	Yes	No
MISS	First Attempt Error	FLOAT	No	0	Yes	Yes	Yes	No
EPOS	Encoder Read Back Position	FLOAT	No	0	Yes	Yes	No	No
MPOS	Motor Position	FLOAT	No	0	Yes	Yes	No	No
ALST	Archive Last Value	FLOAT	No	0	Yes	No	No	No
MLST	Monitor Last Value	FLOAT	No	0	Yes	No	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LVEL	Last Velocity Set	FLOAT	No	0	Yes	Yes	No	No
LACC	Last Acceleration Set	FLOAT	No	0	Yes	Yes	No	No
LVAL	Last Value	SHORT	No	0	Yes	Yes	No	No
CMOD	Current Mode	REC-CHOICE	Yes	0	Yes	Yes	No	No

10. Record Support Routines

init_record

This routine checks to see that device support is available. The routine next checks to see if the device support `sm_command` routine is defined. If either device support or the device support write routine does not exist, an error message is issued and processing is terminated.

If device support includes `init_record`, it is called.

If DOL is a constant, then VAL is initialized with its value and UDF is set to FALSE. If DOL is a PV_LINK then a channel access link is created.

init_sm is then called.

init_sm

The init_sm routine initializes the stepper motor according to its configuration.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_units

Retrieves EGU.

get_precision

Retrieves PREC.

get_graphic_double

Sets the upper display and lower display limits for a field. If the field is VAL, LVAL, MPOS, RBV, EPOS, HIHI, HIGH, LOW, or LOLO, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_control_double

Sets the upper control and the lower control limits for a field. If the field is VAL, LVAL, MPOS, RBV, EPOS, HIHI, HIGH, LOW, or LOLO, the limits are set to DRVH and DRVL. Else if the field has upper and lower limits defined they will be used; else the upper and lower maximum values for the field type will be used.

get_alarm_double

Sets the following values:

upper_alarm_limit = HIHI

upper_warning_limit = HIGH

lower_warning_limit = LOW

lower_alarm_limit = LOLO

11. Record Processing

Not yet written.

12. Device Support

At the present time, device support is intimately connected to record support. The compumotor 1830 and the OMS 6 axis controllers are supported.

Chapter 33:stringin - String Input

1. Introduction

The string input record retrieves an arbitrary ASCII string of up to 40 characters. Several device support routines are available, all of which are soft device support for retrieving values from other records or other software components. Its fields fall into the following categories.

scan parameters

read parameters

operator display parameters

run-time and simulation mode parameters

2. Scan Parameters

The string input record has the standard fields for specifying under what circumstances it will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used.

3. Read Parameters

The INP field determines where the string input record gets its string. It can be a database or channel access link, or a constant. If constant, the VAL field is initialized with the constant and can be changed via dbPuts. Otherwise, the string is read from the specified location each time the record is processed and placed in the VAL field. The maximum number of characters that the string in VAL can be is 40. In addition, the appropriate device support module must be entered into the DTYP field.

See *Address Specification, Chapter 1, 2*, for information on specifying links. The user can see a list of the device support modules currently supported at the user's local site by using the `dbst` utility (R3.13).

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Value	STRING [40]	Yes	Null	Yes	Yes	Yes	Yes
INP	Input Link	INLINK	Yes	0	No	No	N/A	No
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	No

4. Operator Display Parameters

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

5. Alarm Parameters

The string input record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

6. Run-time and Simulation Mode Parameters

The old value field (OVAL) of the string input is used to implement value change monitors for VAL. If VAL is not equal to OVAL, then monitors are triggered.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OVAL	Output Value	STRING	No	Null	Yes	No	No	No

The following fields are used to operate the string input in the simulation mode. See *Simulation Mode, Chapter 3, 2.4*, for more information on simulation mode fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No
SVAL	Simulation Value	STRING [40]	No	Null	Yes	Yes	No	Yes
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

7. Record Support Routines

Three record support routines are provided: init_record, process, and get_value.

init_record

This routine initializes SIMM with the value of SIML if SIML type is CONSTANT link or creates a channel access link if SIML type is PV_LINK. SVAL is likewise initialized if SIOL is CONSTANT or PV_LINK.

This routine next checks to see that device support is available and a record support read routine is defined. If either does not exist, an error message is issued and processing is terminated.

If device support includes init_record, it is called.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

8. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.

2. readValue is called. See *Simulation Mode, Chapter 3, 2.4*, for more information on simulation mode fields and how they affect input.
3. If PACT has been changed to TRUE, the device support read routine has started but has not completed reading a new input value. In this case, the processing routine merely returns, leaving PACT TRUE.
4. TIME is set to tslocaltime
5. Check to see if monitors should be invoked.
   - •Alarm monitors are invoked if the alarm status or severity has changed.
   - •Archive and value change monitors are invoked if OVAL is not equal to VAL.
   - •NSEV and NSTA are reset to 0.
6. Scan forward link if necessary, set PACT FALSE, and return.

9. Device Support

9.1. Fields Of Interest To Device Support

Each stringin input record must have an associated set of device support routines. The primary responsibility of the device support routines is to obtain a new ASCII string value whenever read_stringin is called. The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See Chapter 2, *Fields Common to All Record Types* for an explanation of these fields.
DPVT	DevicePrivate	
UDF	VALUndefined	
VAL	Value	This field is set by the device support routines.
INP	Input Link.	This field is used by the device support routines to locate its input.

9.2. Device Support Routines

Device support consists of the following routines:

report

```
report(FILE fp, paddr)
```

Not currently used.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
     *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

read_stringin

```
read_stringin(precord)
```

This routine must provide a new input value. It returns the following values:

0: Success. A new ASCII string is stored into VAL.

Other: Error.

9.3. Device Support For Soft Records

The `Soft Channel` module places a value directly in VAL.

If the INP link type is constant, the double constant, if non-zero, is converted to a string and stored into VAL by init_record, and UDF is set to FALSE. If the INP link type is PV_LINK, then dbCaAddInlink is called by init_record.

read_stringin calls recGblGetLinkValue to read the current value of VAL. See *Soft Input, Chapter 3, 2.3*.

If the return status of recGblGetLinkValue is zero, then read_stringin sets UDF to FALSE. The status of recGblGetLinkValue is returned.

Chapter 34:stringout — String Output

1. Introduction

The stringout record is used to write an arbitrary ASCII string of up to 40 characters to other records or software variables. Its fields fall into the following categories:

scan parameters

desired output parameters

write parameters

operator display parameters

run-time parameters

2. Scan Parameters

The string output record has the standard fields for specifying under what circumstances it will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used.

3. Desired Output Parameters

The string output record must specify from where it gets its desired output string. The first field that determines where the desired output originates is the output mode select (OSML) field, which can have two possible value: `closed_loop` or `supervisory`. If `supervisory` is specified, DOL is ignored, the current value of VAL is written, and the VAL can be changed externally via dbPuts at run-time. If `closed_loop` is specified, the VAL field's value is obtained from the address specified in the desired output location field (DOL) which can be either a database link or a channel access link.

DOL can also be a constant in addition to a link, in which case VAL is initialized to the constant value. Note that if DOL is a constant, OMSL cannot be `closed_loop`. See *Address Specification, Chapter 1, 2*, for information on specifying links.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Value Field	STRING [40]	Yes	Null	Yes	Yes	Yes	Yes
DOL	Desired Output Location (Input Link)	INLINK	Yes	0	No	No	N/A	No
OMSL	Output Mode Select	GBLCHOICE	Yes	0	Yes	Yes	No	No

4. Write Parameters

The output link specified in the OUT field specifies where the string output record is to write its string. The link can be a database or channel access link. If the OUT field is a constant, no output will be written. See *Address Specification, Chapter 1, 2*, for information on specifying links.

In addition, the appropriate device support module must be entered into the DTYP field. All string output records must specify a device support module. The user can see a list of the device support modules currently supported at the user's local site by using the `dbst` utility (R3.13).

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OUT	Output Link	OUTLINK	Yes	0	No	No	N/A	No
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	No

5. Operator Display Parameters

These parameters are used to present meaningful data to the operator. These fields are used to display the value and other parameters of the string output either textually or graphically.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

The possible alarm conditions for the string output record are the SCAN, READ, and INVALID alarms. The severity of the first two is always MAJOR and not configurable.

The IVOA field specifies an action to take when the INVALID alarm is triggered. There are three possible actions: `Continue normally`, `Don't drive outputs`, and `Set output to IVOV`. When `Set output to IVOV`, the value contained in the IVOV field is written to the output link during an alarm condition. See *Invalid Alarm Output Action, Chapter 3, 3.5* for more information on the IVOA and IVOV fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
IVOA	Invalid Alarm Output Action	GBLCHOICE	Yes	0	Yes	Yes	No	No
IVOV	Invalid Alarm Output Value, in eng. units	DOUBLE	Yes	0	Yes	Yes	No	No

7. Run-time and Simulation Mode Parameters

The old value field (OVAL) of the string input is used to implement value change monitors for VAL. If VAL is not equal to OVAL, then monitors are triggered.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OVAL	Output Value	STRING [40]	No	Null	Yes	No	No	No

The following fields are used to operate the string output in the simulation mode. See *Simulation Mode, Chapter 3, 3.4*, for more information on these fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No
SVAL	Simulation Value	STRING [40]	No	Null	Yes	Yes	No	Yes
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

8. Record Support Routines

Three record support routines are provided: init_record, process, and get_value.

init_record

This routine initializes SIMM if SIML is a constant or creates a channel access link if SIML is PV_LINK. If SIOL is PV_LINK a channel access link is created.

This routine next checks to see that device support is available. The routine next checks to see if the device support write routine is defined. If either device support or the device support write routine does not exist, an error message is issued and processing is terminated.

If DOL is a constant, then the type double constant, if non-zero, is converted to a string and stored into VAL and UDF is set to FALSE. If DOL type is a PV_LINK then dbCaAddInlink is called to create a channel access link.

If device support includes init_record, it is called.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

9. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. If PACT is FALSE and OMSL is CLOSED_LOOP, recGblGetLinkValue is called to read the current value of VAL. See *Soft Output, Chapter 3, 3.2*. If the return status of recGblGetLinkValue is zero then UDF is set to FALSE.
3. Check severity and write the new value. See *Simulation Mode, Chapter 3, 3.4*, and *Invalid Alarm Output Action, Chapter 3, 3.5*, for details on how the simulation mode and the INVALID alarm conditions affect output.
4. If PACT has been changed to TRUE, the device support write output routine has started but has not completed writing the new value. In this case, the processing routine merely returns, leaving PACT TRUE.
5. Check to see if monitors should be invoked.
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - Archive and value change monitors are invoked if OVAL is not equal to VAL.
    - NSEV and NSTA are reset to 0.
6. Scan forward link if necessary, set PACT FALSE, and return.

10. Device Support

10.1. Fields Of Interest To Device Support

Each stringout output record must have an associated set of device support routines. The primary responsibility of the device support routines is to write a new value whenever write_stringout is called. The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active Field	See Chapter 2, *Fields Common to All Record Types* for an explanation of these fields.
DPVT	Device Private	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
VAL	Value	This is the field written by the device support routines.
OUT	Output Link	This field is used by the device support routines to locate its output.

10.2. Device Support Routines

Device support consists of the following routines:

report

```
report(FILE fp, paddr)
```

Not currently used.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
     *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

write_stringout

```
write_stringout(precord)
```

This routine must output a new value. It returns the following values:

- 0: Success.
- Other: Error.

10.3. Device Support for Soft Records

The `Soft Channel` device support module writes the current value of VAL.

If the OUT link type is PV_LINK, then dbCaAddInlink is called by init_record.

write_so calls recGblPutLinkValue to write the current value of VAL. See *Soft Output, Chapter 3, 3.2*.

Chapter 35:subArray

1. Introduction

The normal use for the subArray record type is to obtain sub-arrays from waveform records. Setting either the number of elements (NELM) or index (INDX) fields causes the record to be processed anew so that applications in which the length and position of a sub-array in a waveform record vary dynamically can be implemented using standard EPICS operator interface tools.

The first element of the sub-array, that at location INDX in the referenced waveform record, can be displayed as a scalar, or the entire subarray (of length NELM) can be displayed in the same way as a waveform record. If there are fewer than NELM elements in the referenced waveform after the INDX, only the number of elements actually available are returned, and the number of elements read field (NORD) is set to reflect this. This record type does not support writing new values into waveform records.

The subArray's fields fall into the following categories:

scan parameters

read parameters

array parameters

operator display parameters

run-time parameters

2. Scan Parameters

The subArray record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used.

3. Read Parameters

The subArray's input link (INP) should be configured to reference the Waveform record. It should specify the VAL field of a Waveform record. The INP field can be a channel access link, in addition to a database link. See *Address Specification, Chapter 1, 2*, for information on specifying links.

In addition, the DTYP field must specify a device support module. Currently, the only device support module is `Soft Channel`.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
INP	Input Link	INLINK	Yes	0	No		N/A	No
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	No

4. Array Parameters

These parameters determine the number of array elements (the array length) and the data type of those elements. The Field Type of Value (FTVL) field determines the data type of the array.

The user specifies the maximum number of elements allowed in the subarray in the MALM field. Generally, the number should be equal to the number of elements of the Waveform array (found in the Waveform's NELM field). The MALM field is used to allocate memory. The subArray's Number of Elements (NELM) field is where the user specifies the actual number of elements that the subArray will contain. It should of course be no greater than MALM; if it is, the record processing routine sets it equal to MALM.

The INDX field determines the offset of the subArray record's array in relation to the Waveform's. For instance, if INDX is 2, then the subArray will read NELM elements starting with the third element of the Waveform's array. Thus, it equals the index number of the Waveform's array.

The actual sub-array is referenced by the VAL field.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
FTVL	Field Type of Value	GBLCHOICE	Yes	0	Yes	No	No	No
VAL	Value Field	(See FTVL)	No	0	Yes	Yes	Yes	No
MALM	Maximum Number of Elements In Sub-array	ULONG	Yes	1	Yes	No	No	No
NELM	Number of Elements In Sub-array	ULONG	Yes	1	Yes	Yes	No	Yes
INDX	Index Into Referenced Array	ULONG	Yes	0	Yes	Yes	No	Yes

5. Operator Display Parameters

These parameters are used to present meaningful data to the operator. They display the value and other parameters of the subarray record either textually or graphically.

EGU is a string of up to 16 characters describing the engineering units (if any) of the values which the subArray holds. It is retrieved by the `get_units` record support routine.

The HOPR and LOPR fields set the upper and lower display limits for the sub-array elements. Both the `get_graphic_double` and `get_control_double` record support routines retrieve these fields.

The PREC field determines the floating point precision with which to display VAL. It is used whenever the `get_precision` record support routine is called.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	null	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

The subarray record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

7. Run-time Parameters

These fields are not configurable by the user. They are used for the record's internal processing or to represent the current state of the record.

The NORD field holds a counter of the number of elements read into the array. It can be less than NELM even after the array is full if NELM exceeds the number of existing elements in the referenced array, i.e., the Waveform's array.

BPTR contains a pointer to the record's array.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
NORD	Number of Elements Read	LONG	No	0	Yes	No	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
BPTR	Buffer Pointer	NOACCESS	No	null	Yes	No	No	No

8. Record Support Routines

init_record

Using MALM and FTVL, space for the array is allocated. The array address is stored in BPTR. This routine checks to see that device support is available and a device support read routine is defined. If either does not exist, an error message is issued and processing is terminated. If device support includes init_record, it is called.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to the sub-array.

cvt_dbaddr

This is called by dbNameToAddr. It makes the dbAddr structure refer to the actual buffer holding the result.

get_array_info

Retrieves NELM.

put_array_info

Sets NORD.

get_graphic_double

For the elements in the array, this routine routines HOPR and LOPR. For the INDX field, this routine returns MALM - 1 and 0. For NELM, it returns MALM and 1. For other fields, it calls `recGblGetGraphicDouble()`.

get_control_double

For array elements, this routine retrieves HOPR and LOPR. Otherwise, `recGblGetControlDouble()` is called.

get_units

Retrieves EGU.

get_prec

Retrieves PREC.

9. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. Sanity check NELM and INDX. If NELM is greater than MALM it is set to MALM. If INDX is greater than MALM it is set to MALM-1.
3. Call device support read routine. This routine is expected to place the desired sub-array at the beginning of the buffer and set NORD to the number of elements of the sub-array that were read.
4. If PACT has been changed to TRUE, the device support read routine has started but has not completed writing the new value. In this case, the processing routine merely returns, leaving PACT TRUE. Otherwise, process sets PACT TRUE at this time. This asynchronous processing logic is not currently used but has been left in place.
5. Check to see if monitors should be invoked.
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - Archive and value change monitors are always invoked.
    - NSEV and NSTA are reset to 0.
6. Scan forward link if necessary, set PACT FALSE, and return.

10. Device Support

10.1. Fields Of Interest To Device Support

The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See Chapter 2, *Fields Common to All Record Types* for an explanation of these fields.
DPVT	Device Private	
UDF	VAL Undefined	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
INP	Input Link	This field is used by the device support routines to locate its input.
FTVL	Field Type of Value	This is DBF_STRING, ... ,DBF_ENUM. The device support routine should check that this is correctly defined.
MALM	Maximum Number Of Elements in Sub-array	Number of elements that will fit in the array the record allocates. Device support must never return more elements than this.
NELM	Number Sub-array Elements	Number of elements in desired sub-array.
INDX	Index Into Referenced Array	Index of beginning of desired sub-array in source array.
BPTR	Buffer Pointer	Address of array device support must copy the source array into.
NORD	Number Of Elements Read	Device support must set this value when it completes.

10.2. Device Support Routines

Device support consists of the following routines:

report

```
report(FILE fp, paddr)
```

not currently used.

init

```
init()
```

Not currently used.

init_record

```
init_record(precord)
```

This routine is called by the record support init_record routine.

read_sa

```
read_sa(precord)
```

Enough of the source waveform is read into BPTR, from the beginning of the source, to include the requested sub-array. The sub-array is then copied to the beginning of the buffer. NORD is set to indicate how many elements of the sub-array were acquired.

10.3. Device Support For Soft Records

Only the device support module `Soft Channel` is currently provided. The INP link type must be either DB_LINK or CA_LINK.

Soft Channel

INP is expected to point to a waveform record.

Chapter 36:sub - Subroutine

1. Introduction

The subroutine record is used to call a C initialization routine and a recurring scan routine. There is no device support for this record. The fields fall into several categories:

scan parameters

read parameters

subroutine connection parameters

operator display parameters

alarm parameters

monitor parameters

run-time parameters.

2. Scan Parameters

The subroutine record has the standard fields for specifying under what circumstances it will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used.

3. Read Parameters

The subroutine record has twelve input links (INPA-INPL), each of which has a corresponding value field (A-L). These fields are used to retrieve and store values that can be passed to the subroutine that the record calls.

The input links can be either channel access or database links, or constants. When constants, the corresponding value field for the link is initialized with the constant value and the field's value can be changed at run-time via dbPuts. Otherwise, the values for (A-F) are fetched from the input links when the record is processed. See *Address Specification, Chapter 1, 2*, for information on specifying links.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
INPA	Input A	INLINK	Yes	0	No	No	N/A	No
INPB	Input B	INLINK	Yes	0	No	No	N/A	No
INPC	Input C	INLINK	Yes	0	No	No	N/A	No
INPD	Input D	INLINK	Yes	0	No	No	N/A	No
INPE	Input E	INLINK	Yes	0	No	No	N/A	No
INPF	Input F	INLINK	Yes	0	No	No	N/A	No
INPG	Input G	INLINK	Yes	0	No	No	N/A	No
INPH	Input H	INLINK	Yes	0	No	No	N/A	No
INPI	Input F	INLINK	Yes	0	No	No	N/A	No
INPJ	Input H	INLINK	Yes	0	No	No	N/A	No
INPK	Input K	INLINK	Yes	0	No	No	N/A	No
INPL	Input L	INLINK	Yes	0	No	No	N/A	No
A	Value for A	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
B	B Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
C	C Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
D	D Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
E	E Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
F	F Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
G	G Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
H	H Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
I	I Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
J	J Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
K	K Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes
L	L Value	DOUBLE	No	0	Yes	Yes/No	Yes	Yes

4. Subroutine Connection

These fields are used to connect to the C subroutine. The name of the subroutine should be entered in the SNAM field.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
INAM	Initialization Name	STRING [16]	Yes	Null	Yes	No	No	No
SNAM	Subroutine Name	STRING [16]	Yes	Null	Yes	No	No	No

5. Operator Display Parameters

These parameters are used to present meaningful data to the operator. They display the value and other parameters of the subroutine either textually or graphically.

EGU is a string of up to 16 characters that could describe any units used by the subroutine record. It is retrieved by the `get_units` record support routine.

The HOPR and LOPR fields set the upper and lower display limits for the VAL, A-L, LA-LL, HIHI, LOLO, LOW, and HIGH fields. Both the `get_graphic_double` and `get_control_double` record support routines retrieve these fields.

The PREC field determines the floating point precision with which to display VAL. It is used whenever the `get_precision` record support routine is called.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	null	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

The possible alarm conditions for subroutine records are the SCAN, READ, limit alarms, and an alarm that can be triggered if the subroutine returns a negative value. The SCAN and READ alarms are called by the record or device support routines. The limit alarms are configured by the user in the HIHI, LOLO, HIGH, and LOW fields using numerical values. They apply to the VAL field. For each of these fields, there is a corresponding severity field which can be either NO_ALARM, MINOR, or MAJOR.

The BRSV field is where the user can set the alarm severity in case the subroutine returns a negative value. See *Alarm Specification, Chapter 1, 4*, for a complete explanation of alarms and these fields. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HIHI	Hihi Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HIGH	High Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOW	Low Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
LOLO	Lolo Alarm Limit	FLOAT	Yes	0	Yes	Yes	No	Yes
HHSV	Hihi Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HSV	High Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LSV	Low Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
LLSV	Lolo Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
BRSV	Severity for a subroutine return value less than 0.	GBLCHOICE	Yes	0	Yes	Yes	No	Yes
HYST	Alarm Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

7. Monitor Parameters

These parameters are used to determine when to send monitors placed on the VAL field. The appropriate monitors are invoked when VAL differs from the values in the ALST and MLST run-time fields, i.e., when the value of VAL changes by more than the deadband specified in these fields. The ADEL and MDEL fields specify a minimum delta which the change must surpass before the value-change monitors are invoked. If these fields have a value of zero, everytime the value changes, a monitor will be triggered; if they have a value of -1, everytime the record is processed, monitors are triggered. The ADEL field is used by archive monitors and the MDEL field for all other types of monitors. See *Monitor Specification, Chapter 1, 5*, for a complete explanation of monitors and deadbands.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ADEL	Archive Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
MDEL	Monitor, i.e. value change, Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

8. Run-time Parameters

These parameters are used by the run-time code for processing the subroutine record. They are not configured using a database configuration tool. They represent the current state of the record. Many of them are used by the record processing routines or the monitors.

VAL should be set by the subroutine. SADR holds the subroutine address and is set by the record processing routine. STYP, the subroutine symbol type, is also set by the record processing routine.

The rest of these fields—LALM, ALST, MLST, and the LA-LL fields—are used to implement the monitors. For example, when LA is not equal to A, the value-change monitors are called for that field.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Value Field	DOUBLE	No	0	Yes	Yes	Yes	Yes
SADR	Subroutine Address	NOACCESS	No	0	No	No	No	
STYP	Subroutine Symbol Type	SHORT	No	0	Yes	No	No	No
LALM	Last Alarm Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No
ALST	Last Archiver Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No
MLST	Last Value Change Monitor Trigger Value	DOUBLE	No	0	Yes	No	No	No
LA	Last A Value	DOUBLE	No	0	Yes	No	No	No
LB	Last B Value	DOUBLE	No	0	Yes	No	No	No
LC	Last C Value	DOUBLE	No	0	Yes	No	No	No
LD	Last D Value	DOUBLE	No	0	Yes	No	No	No
LE	Last E Value	DOUBLE	No	0	Yes	No	No	No
LF	Last F Value	DOUBLE	No	0	Yes	No	No	No
LG	Last G Value	DOUBLE	No	0	Yes	No	No	No
LH	Last H Value	DOUBLE	No	0	Yes	No	No	No
LI	Last I Value	DOUBLE	No	0	Yes	No	No	No
LJ	Last J Value	DOUBLE	No	0	Yes	No	No	No
LK	Last K Value	DOUBLE	No	0	Yes	No	No	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LL	Last L Value	DOUBLE	No	0	Yes	No	No	No

9. Record Support Routines

init_record

For each constant input link, the corresponding value field is initialized with the constant value. For each input link that is of type PV_LINK, a channel access link is created.

If an initialization subroutine is defined, it is located and called.

The processing subroutine is located and its address and type stored in SADR and STYP.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to VAL.

get_units

Retrieves EGU.

get_precision

Retrieves PREC when VAL is the field being referenced. Otherwise, calls `recGblGetPrec()`.

get_graphic_double

Sets the upper display and lower display limits for a field. If the field is VAL, A-L, LA-LL, HIHI, HIGH, LOW, or LOLO, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_control_double

Sets the upper control and the lower control limits for a field. If the field is VAL, A-L, LA-LL, HIHI, HIGH, LOW, or LOLO, the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_alarm_double

Sets the following values:

upper_alarm_limit = HIHI

upper_warning_limit = HIGH

lower_warning_limit = LOW

lower_alarm_limit = LOLO

10. Record Processing

Routine process implements the following algorithm:

1. If PACT is FALSE then fetch all arguments.
2. Call the subroutine and check return value.
    - Call subroutine
    - Set PACT TRUE
    - If return value is 1, return
3. Check alarms. This routine checks to see if the new VAL causes the alarm status and severity to change. If so, NSEV, NSTA and LALM are set. It also honors the alarm hysteresis factor (HYST). Thus the value must change by more than HYST before the alarm status and severity is lowered.
4. Check to see if monitors should be invoked.
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - Archive and value change monitors are invoked if ADEL and MDEL conditions are met.
    - Monitors for A-L are invoked if value has changed.
    - NSEV and NSTA are reset to 0.
5. Scan forward link if necessary, set PACT FALSE, and return.

11. Example Synchronous Subroutine

This is an example that merely increments VAL each time process is called.

```
#include        <vxWorks.h>
#include        <types.h>
#include        <stdioLib.h>
#include        <dbDefs.h>
#include        <subRecord.h>
#include        <dbCommon.h>
#include        <recSup.h>
long subInit(psub)
    struct subRecord     *psub;
{
    printf("subInit was called\n");
    return(0);
}
```

```
long subProcess(psub)
    struct subRecord     *psub;
{
    psub->val++;
    return(0);
}
```

12. Example Asynchronous Subroutine

This example shows an asynchronous subroutine. It uses (actually misuses) fields A and B. Field A is taken as the number of seconds until asynchronous completion. Field B is a flag to decide if messages should be printed. Lets assume A > 0 and B = 1. The following sequence of actions will occcur:

1. subProcess is called with pact FALSE. It performs the following steps.
   b. Computes, from A, the number of ticks until asynchronous completion should occur.
   c. Prints a message stating that it is requesting an asynchronous callback.
   d. Calls the vxWorks watchdog start routine.
   e. Sets pact TRUE and returns a value of 0. This tells record support to complete without checking alarms, monitors, or the forward link.
6. When the time expires, the system wide callback task calls myCallback. myCallback locks the record, calls process, and unlocks the record.
7. Process again calls subProcess, but now pact is TRUE. Thus the following is done:
   h. VAL is incremented.
   i. A completion message is printed.
   j. subProcess returns 0. The record processing routine will complete record processing.

```
#include   <vxWorks.h>
#include   <types.h>
#include   <stdioLib.h>
#include   <wdLib.h>
#include   <callback.h>
#include   <dbDefs.h>
#include   <dbAccess.h>
#include   <subRecord.h>
/* control block for callback*/
struct callback {
  CALLBACK   callback;
  struct dbCommon *precord;
  WDOG_ID wd_id;
};

void myCallback(pcallback)
    struct callback *pcallback;
{
    struct dbCommon *precord=pcallback->precord;
    struct rset  *prset=(struct rset *)(precord->rset);
    dbScanLock(precord);
    (*prset->process)(precord);
    dbScanUnlock(precord);
```

```
}

long subInit(psub)
    struct subRecord  *psub;
{
    struct callback *pcallback;
    pcallback = (struct callback *)(calloc(1,sizeof(struct
      callback)));
    psub->dpvt = (void *)pcallback;
    callbackSetCallback(myCallback,pcallback);
    pcallback->precord = (struct dbCommon *)psub;
    pcallback->wd_id = wdCreate();
    printf("subInit was called\n");
    return(0);
}

long subProcess(psub)
    struct subRecord  *psub;
{
    struct callback *pcallback=(struct callback *)(psub->dpvt);
    int  wait_time;
    /* sub.inp must be a CONSTANT*/
  if(psub->pact) {
    psub->val++;
    if(psub->b)
      printf("%s subProcess Completed\n",psub->name);
      return(0);
  } else {
    wait_time = (long)(psub->a * vxTicksPerSecond);
    if(wait_time<=0){
      if (psub->b)
        printf("%s subProcess synchronous processing\n",psub-
      >name);
        psub->pact = TRUE;
        return(0);
    }
    if (psub->b){
      callbackSetPriority(psub->prio,pcallback);
      printf("%s Starting asynchronous processing\n",psub-
      >name);
      wdStart(pcallback-
      >wd_id,wait_time,callbackRequest,(int)pcallback);
      return(1);
    }
  }
    return(0);
}
```

Chapter 37:Timer

1. Introduction

The timer record is provided as a means for configuring timing outputs. These records will drive an output that may be used to latch data into analog inputs or waveform digitizers. The fields for the timer record fall into the following categories:

scan parameters

setup parameters

write and convert parameters

operator display parameters

event generation parameters

run-time parameters

2. Scan Parameters

The timer record has the standard fields for specifying under what circumstances it will be processed. In addition, the VAL field is provided in the timer record to force record processing when written to. The other fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Value Field	SHORT	No	0	Yes	Yes	Yes	Yes

3. Setup Parameters

The timer record has fields needed to initialize a motor. The trigger source (TRSC) field is usually configured to come in from an external signal. This is configured in the record by setting TSRC to `external`, the only other choice being `internal`. The pre-trigger state field (PSRC) is the state of the timing channel when it is not fired. The timing channel usually goes high when the timing pulse is present. In such a case the pre-trigger state is low.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
TSRC	Clock Source	RECCHOICE	Yes	0	Yes	Yes	No	No
PTST	Pre-Trigger State	RECCHOICE	Yes	0	Yes	Yes	No	Yes

4. Write and Convert Parameters

The delays and outputs are specified in the time units (TIMU), which has four choices based on seconds: `Milliseconds`, `Microseconds`, `Nanoseconds`, `and Picoseconds`.

There are delays and pulse widths for five timing pulses per channel. The pulse width for each should be specified in the OPW1-OPW5 fields, in the units chosen in the TIMU field. The delay width for each trigger is specified in the DUT1-DUT5 fields. The delay width is the time in between pulses. These fields are converted to seconds by the run-time code before being used.

The value of the trigger delay field (TRDL) fetched from the trigger origin (TORG) field is added to the delay for each of these five signals. The TORG and TRDL work like the desired output parameters of other output records. TORG can be a channel access or database link, or a constant. If TORG is a constant, TRDL is initialized to the constant value and can be changed at run-time via dbPuts. Using the TORG field one can set up timing chains which are relative to other timing signals. See *Address Specification, Chapter 1, 2*, for information on how to specify database links.

The desired output is sent to the address contained in the output link (OUT). The DTYP field must contain the name of a valid device support module. The user can see a list of the device support modules currently supported at the user's local site by using the `dbst` utility (R3.13).

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
TORG	Trigger Delay Origin (input link)	INLINK	Yes	0	No	No	N/A	No
TRDL	Trigger Delay	FLOAT	No	0	Yes	Yes	No	No
TIMU	Timer Units	RECCHOICE	Yes	0	Yes	Yes	No	No
DTYP	Device Type	DEVCHOICE	Yes	0	Yes	No	No	No
OUT	Output Link	OUTLINK	Yes	0	No	No	N/A	No
DUT1	Delay Width for trigger 1	FLOAT	Yes	0	Yes	Yes	No	Yes
OPW1	Pulse width for Trigger 1	FLOAT	Yes	0	Yes	Yes	No	Yes
DUT2	Delay Width for Trigger 2	FLOAT	Yes	0	Yes	Yes	No	Yes

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OPW2	Pulse Width for Trigger 2	FLOAT	Yes	0	Yes	Yes	No	Yes
DUT3	Delay Width for Trigger 3	FLOAT	Yes	0	Yes	Yes	No	Yes
OPW3	Pulse Width for Trigger 3	FLOAT	Yes	0	Yes	Yes	No	Yes
DUT4	Delay Width for Trigger 4	FLOAT	Yes	0	Yes	Yes	No	Yes
OPW4	Pulse Width for Trigger 4	FLOAT	Yes	0	Yes	Yes	No	Yes
DUT5	Delay Width for Trigger 5	FLOAT	Yes	0	Yes	Yes	No	Yes
OPW5	Pulse Width for Trigger 5	FLOAT	Yes	0	Yes	Yes	No	Yes

5. Operator Display Parameters

The PDLY field displays the delay from source to input, which is the time is takes for the timing signal to get from its output to the trigger. It is not used by the run-time code, but characterizes the timing signals for the operator’s knowledge.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
PDLY	Delay Source to Input	FLOAT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

6. Alarm Parameters

The timer record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

7. Event Generation Parameters

The timing event (TEVT) field is used to generate internal events for processing records on an event. It should be an integer representing an event number.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
TEVT	Event Number To Be Posted On Trigger.	SHORT	Yes	0	Yes	Yes	No	Yes

8. Run-time Parameters

These parameters are used by the run-time code for processing the timing channel. They are not configured using a configuration tool. They represent the current state of the timing channel.

The T1DL-T5DL and T1WD-T5WD fields are where the record stores the user-configured delay widths and pulse widths for the five triggers (DUT1-DUT5 and OPW1-OPW5) after it has converted them to seconds.

The T1TD-T5TD and T1LD-T5LD fields hold the trailing and leading trigger delays for each trigger. These fields are calculated by the record as follows:

T*n*LD = DUT*n* + TRDL

T*n*TD = T*n*LD + OPW*n*

—where *n* represents the trigger number.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
T1DL	Delay Width for Trigger 1	DOUBLE	No	0	Yes	Yes	No	No
T1WD	Pulse Width for Trigger 1	DOUBLE	No	0	Yes	Yes	Yes	No
T2DL	Delay Width for Trigger 2	DOUBLE	No	0	Yes	Yes	No	No
T2WD	Pulse Width for Trigger 2	DOUBLE	No	0	Yes	Yes	No	No
T3DL	Delay Width for Trigger 3	DOUBLE	No	0	Yes	Yes	No	No
T3WD	Pulse Width for Trigger 3	DOUBLE	No	0	Yes	Yes	No	No
T4DL	Delay Width or Trigger 4	DOUBLE	No	0	Yes	Yes	No	No
T4WD	Pulse Width for Trigger 4	DOUBLE	No	0	Yes	Yes	No	No
T5DL	Delay width for Trigger 5	DOUBLE	No	0	Yes	Yes	No	No
T5WD	Pulse Width for Trigger 5	DOUBLE	No	0	Yes	Yes	No	No
T1TD	Trigger 1 Trailing Delay	FLOAT	No	0	Yes	Yes	Yes	No
T1LD	Trigger 1 Leading Delay	FLOAT	No	0	Yes	Yes	Yes	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
T2TD	Trigger 2 Trailing Delay	FLOAT	No	0	Yes	Yes	No	No
T2LD	Trigger 2 Leading Delay	FLOAT	No	0	Yes	Yes	No	No
T3TD	Trigger 3 Trailing Delay	FLOAT	No	0	Yes	Yes	No	No
T3LD	Trigger 3 Leading Delay	FLOAT	No	0	Yes	Yes	No	No
T4TD	Trigger 4 Trailing Delay	FLOAT	No	0	Yes	Yes	No	No
T4LD	Trigger 4 Leading Delay	FLOAT	No	0	Yes	Yes	No	No
T5TD	Trigger 5 Trailing Delay	FLOAT	No	0	Yes	Yes	No	No
T5LD	Trigger 5 Leading Delay	FLOAT	No	0	Yes	Yes	No	No
TDIS	Timing Pulse Disable	SHORT	No	0	Yes	Yes	No	Yes
MAIN	Maintain on Reboot	GBLCHOICE	Yes	1	Yes	Yes	No	Yes
RDT1	Reboot Delay of 1	FLOAT	No	0	Yes	Yes	No	No
RDW1	Reboot Width of 1	FLOAT	No	0	Yes	Yes	No	No

Chapter 38: Wait

Ned D. Arnold
Advanced Photon Source
Argonne National Laboratory

1. Introduction

This chapter describes the capabilities and use of the Wait record. The Wait record was developed primarily to provide "reassignable" link fields that could be changed during run time. A unique library was developed to provide this feature. Since all EPICS standard link fields are "reassignable" in R3.13 and above, the Wait record and its associated library are somewhat obsolete. For new designs, it is recommended that the *Calcout* record be used, which provides the same function as the Wait record (plus more) while using the standard EPICS links.

The Wait record is derived from the Calc record with the following additional features:

- reassignable PV links
- an output link
- a desired output link
- an output event number to post
- and several options as to when it will execute the output link and event posting.

The Wait record also has the capability to process as a result of an input changing (via CA monitors), and it can be used to do *conditional* processing within the database. Its name is derived from the original requirement that initiated its development— "I want to wait until all the motors have stopped and then trigger the detector." Its fields fall into the following categories:

scan parameters

read parameters

expression-related parameters

desired output parameters

write parameters

operator parameters

monitor parameters

run-time and simulation mode parameters

2. Scan Parameters

The wait record has the standard fields for specifying under what circumstances the record will be processed. In addition to the standard scan mechanisms available to all records (periodic, passive, event, etc.), the wait record can be specified to process as a result of one of its input values changing (using Channel Access monitors). This offers immediate response to an input change (rather than waiting for the next periodic scan) while minimizing record processing that is not required. The scan mechanism choice `I/O Intr` will enable this feature. For the current EPICS release, an additional module, `caMonitor.o`, must be compiled and loaded with the wait record support to provide this feature.

> **Note: Because of the event driven nature of this feature, it is quite easy to configure a database that results in an infinite loop which uses all available CPU time. If the wait record is set to process as a result of a channel changing and the processing of the wait record causes the channel to change again, an infinite loop will result. The symptom will be a loss of all channel access connections (lower priority tasks) even though the shell responds normally. Using the `spy` vxWorks utility will confirm the predicament by showing 0% free CPU time.**

Scanning Specification, Chapter 1, 1, explains how the rest of these fields are used. They are listed in *Scan Fields, Chapter 2, 2*.

3. Read Parameters

Like the Calc record, the read parameters for the Wait record consist of 12 input links—INAN, INBN, . . . INLN. The fields can be database links or constants. If they are links, they must specify another record's field in the same IOC. If they are constants, the corresponding value field for the link (A-L) will be initialized with the constant's value and can be changed via dbPuts. The Wait record does not support direct links to hardware.

Unlike the Calc record, these input links can be modified during run time. These are ASCII fields which have a special processing routine attached to them. When changed, the routine is called. The record will use the new link the next time the record is processed.

A consequence of reassignable links is that one cannot force the processing of any specified records prior to retrieving the data from them (i.e. there is no .PP flag). This should be considered when designing a database using the Wait record.

In this initial version, the *reassignable* links do not support channel access connections external to the IOC. Until this feature is added, the specified Process Variables must reside on the same IOC.

See *Address Specification, Chapter 1, 2*, for information on specifying links and constants.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor
INAN	Input Link A Name	STRING[40]	Yes	0	Yes	Yes	Yes
INBN	Input Link B Name	STRING[40]	Yes	0	Yes	Yes	Yes
INCN	Input Link C Name	STRING[40]	Yes	0	Yes	Yes	Yes
INDN	Input Link D Name	STRING[40]	Yes	0	Yes	Yes	Yes

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor
INEN	Input Link E Name	STRING[40]	Yes	0	Yes	Yes	Yes
INFN	Input Link F Name	STRING[40]	Yes	0	Yes	Yes	Yes
INGN	Input Link G Name	STRING[40]	Yes	0	Yes	Yes	Yes
INHN	Input Link H Name	STRING[40]	Yes	0	Yes	Yes	Yes
ININ	Input Link I Name	STRING[40]	Yes	0	Yes	Yes	Yes
INJN	Input Link J Name	STRING[40]	Yes	0	Yes	Yes	Yes
INKN	Input Link K Name	STRING[40]	Yes	0	Yes	Yes	Yes
INLN	Input Link L Name	STRING[40]	Yes	0	Yes	Yes	Yes

4. Expression-related Parameters

The expression related fields of the Wait record are identical to that of the Calc record. The two main fields are the CALC and RPCL fields. The CALC field contains the infix expression which the record routine will use when it processes the record. The user enters an expression in this field. The resulting value of the expression is placed in the VAL field The write parameters of the record can write this value if configured to.

The CALC field is actually converted to opcode and stored as a reverse polish expression, or postfix expression, in the RPCL field. It is this expression which is actually used to calculate VAL. CALC can be changed at run-time, and the special record routine calls a function to convert it to the postfix expression. One difference between the Calc record and the Wait record is that the Wait record contains a flag field (CLCV) that indicates if the expression is a valid one. It will be true (non-zero) if the expression is invalid, in which case an error message is generated.

The reverse polish calculation is evaluated most efficiently during run-time. The range of expressions supported by the calculation record are separated into operands, algebraic operations, trigonometric operations, relational operations, logical operations, parentheses and commas, and the question mark operator.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
CALC	Calculation	DBF_STRING	Yes	0	Yes	Yes	Yes	Yes
RPCL	Reverse Polish	DBF_NOACCESS	No	0	No	No	N/A	No

4.1. Operands

The expression can use the values retrieved from the INP*x* links as operands, these values being stored in the A-L fields. The values to be used in the expression are simply referenced by the field letter. For instance, the value obtained from the INPA link is stored in field A and the value obtained from INPB is stored in field B. The field names can be included in the expression which will operate on their respective values, as in A+B. In addition, the RNDM unary function can be included as an operand to generate a random number between 0 and 1.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor
A	Input Values A	DOUBLE	No	0	Yes	Yes/No	Yes
B	Input Values B	DOUBLE	No	0	Yes	Yes/No	Yes
C	Input Values C	DOUBLE	No	0	Yes	Yes/No	Yes
D	Input Values D	DOUBLE	No	0	Yes	Yes/No	Yes
E	Input Values E	DOUBLE	No	0	Yes	Yes/No	Yes
F	Input Values E	DOUBLE	No	0	Yes	Yes/No	Yes
G	Input Values G	DOUBLE	No	0	Yes	Yes/No	Yes
H	Input Values H	DOUBLE	No	0	Yes	Yes/No	Yes
I	Input Values I	DOUBLE	No	0	Yes	Yes/No	Yes
J	Input Values J	DOUBLE	No	0	Yes	Yes/No	Yes
K	Input Values K	DOUBLE	No	0	Yes	Yes/No	Yes
L	Input Values L	DOUBLE	No	0	Yes	Yes/No	Yes

4.2. Algebraic Operators

ABS: Absolute value (unary)
SQR: Square root (unary)
MIN: Minimum (binary function)
MAX: Maximum (binary function)
CEIL: Ceiling (unary)
FLOOR: Floor (unary)
LOG: Log base 10 (unary)
LOGE: Natural log (unary)
EXP: Exponential function (unary)
^ : Exponential (binary)
** : Exponential (binary)
+ : Addition (binary)
- : Subtraction (binary)
* : Multiplication (binary)

/ : Division (binary)
% : Modulo (binary)
NOT: Negate (unary)

4.3. Trigonometric Operators

SIN: Sine
SINH: Hyperbolic sine
ASIN: Arc sine
COS: Cosine
COSH: Hyperbolic cosine
ACOS: Arc cosine
TAN: Tangent
TANH: Hyperbolic tangent
ATAN: Arc tangent

4.4. Relational Operators

>= : Greater than or equal to
> : Greater than
<= : Less than or equal to
<: Less than
: Not equal to
= : Equal to

4.5. Logical Operators

&& : And
|| : Or
! : Not

4.6. Bitwise Operators

| : Bitwise Or
& : Bitwise And
OR : Bitwise Or
AND: Bitwise And
XOR: Bitwise Exclusive Or
~ : One's Complement

<< : Left shift

>> : Right shift

4.7. parentheses and Comma

The open and close parentheses are supported. Nested parentheses are supported.

The comma is supported when used to separate the arguments of a binary function.

4.8. Conditional Expression

The “C” question mark operator is supported. The format is:

```
(condition)? True result : False result
```

4.9. Examples

Algebraic

```
A + B
```

•Result is A + B

Relational

```
(A + B) < (C + D)
```

•Result is 1 if (A+B) < (C+D)

•Result is 0 if (A+B) >= (C+D)

Question Mark

```
(A+B)<(C+D)?E:F
```

•Result is E if (A+B) < (C+D)

•Result is F if (A+B) >= (C+D)

```
(A+B)<(C+D)?E
```

•Result is E if (A+B) < (C+D)

•Result is unchanged if (A+B) >= (C+D)

Logical

```
A&B
```

•Causes the following to occur:

- Convert A to integer
- Convert B to integer

- Bit-wise And A and B
- Convert result to floating point

5. Desired Output Parameters

Using the choices in the Data Output field (DOPT) explained in the next section, the Wait record can be configured to write as its output the result of its calculation (the value in VAL) or it can be configured to write a desired output value, fetched from an input link. The desired output fields for the Wait record are similar to the desired output parameters of other output records.

In the desired output location (DOLN) field, the user specifies a link from which the record will retrieve the value that it writes (if the DOPT field specifies DOLD). The value is retrieved and placed in the desired output location data field (DOLD). If DOLN specifies no link, a flag field indicates that the DOLN field is invalid. When the DOLN field is invalid, the value of DOLD can be set via dbPuts at run-time.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
DOLN	Desired Output Location	STRING [40]	Yes	Null	Yes	Yes	No	No
DOLD	Desired Output Location Data	DOUBLE	Yes	0	Yes	Yes	Yes	No

6. Write Parameters

The Wait record can write either the result of its calculation (VAL) or the value retrieved by the DOLN field. If the user specifies `Use VAL` in the Data Option (DOPT) field, then the value in VAL will be written. If the user specifies `Use DOL`, the value retrieved from the DOLN link and contained in the DOLD field will be written. The DOPT field can be changed during run-time.

The Output Link Name (OUTN) specifies where the record is to write the output value.

The Wait record does not have to write its output every time the record processes. This allows “downstream” processing of records to be done conditionally. The record has *output execution* options which can be specified in the OOPT field. These options are as follows.

- `Every Time`: Outputs are executed every time the record processes.
- `On Change`: Outputs are executed if the result of the calculation (VAL) is different than the previous time the record was processed.
- `When Zero`: Outputs are executed if the result of the calculation is zero.
- `When Non-zero`: Outputs are executed if the result of the calculation is not zero.
- `Transition To Zero`: Outputs are executed if the result of the calculation is zero and the previous value was not zero.
- `Transition To Non-zero`: Outputs are executed if the result of the calculation is not zero and the previous value was zero.

The Wait record can also post an event. If a non-zero value is entered into the OEVT (Output Event) field, the record will "post an event" (using the entered number as the event number) whenever the output link is executed. This is a way of initiating several other records to process as a result of a calculation.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
OOPT	Output Option	RECCHOICE	Yes	0	Yes	Yes	No	No
OUTN	Output Link Name	STRING [40]	Yes	Null	Yes	Yes	No	No
DOPT	Data Option	RECCHOICE	Yes	0	Yes	Yes	No	No
OEVT	Output Event	USHORT	Yes	0	Yes	Yes	No	No

7. Operator Display Parameters

These parameters are used to present meaningful data to the operator. They display the value and other parameters of the Wait record either textually or graphically.

The HOPR and LOPR fields set the upper and lower display limits for the VAL field. The `get_graphic_double` retrieves these fields.

The PREC field determines the floating point precision with which to display VAL. It is used whenever the `get_precision` record support routine is called.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

8. Alarm Parameters

The wait record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

9. Monitor Parameters

These parameters are delta values that implement a deadband on the value-change monitors for the VAL field. Monitors are sent when the value field exceeds the last monitored field (MLST, for instance) by the appropriate delta. If these fields have a value of zero, everytime the value changes, a monitor will be triggered; if they have a value of -1, everytime the record is processed, monitors are triggered. The ADEL field is used for archive monitors and the MDEL field for all other types of monitors. See *Monitor Specification, Chapter 1, 5*, for a complete explanation of monitors.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
ADEL	Archive Deadband	DOUBLE	Yes	0	Yes	Yes	No	No
MDEL	Monitor, i.e. value change, Deadband	DOUBLE	Yes	0	Yes	Yes	No	No

10. Run-time Parameters

These parameters are used by the run-time code for processing the Wait record. They are not configurable. They represent the current state of the record. The record support routines use some of them to process the record or to implement monitors.

The OVAL field is used to implement monitors on the VAL field. If VAL differs from OVAL (taking into account the deadband) then monitors for VAL are triggered.

The LA-LL fields are used to implement monitors for the A-L field, the fields which hold the values retrieved from the input links. They hold the values retrieved from INPA-INPL the last time the record was processed. For instance, when A, the current value retrieved from INPA, does not equal LA, the last value retrieved, then monitors for A are triggered.

The DOLV and CLCV are flag fields. The DOLV flag is used to indicate if the process variable specified in the Desired Output Location (DOLN) field is a valid name. If it is not, the desired output value is not retrieved from DOLN. The CLCV flag is used to indicate if the Wait record's CALC expression is a mathematically valid one.

The CBST field contains a pointer to a record private structure. It is not of concern to the user.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
CBST	Callback Structure	NOACCESS	No	Null	No	No	No	No
OVAL	Old Value	DOUBLE	No	0	Yes	Yes	No	No
DOLA	Desired Output Location Address	NOACCESS	No	Null	No	No	No	No
DOLV	Desired Output Location Valid	LONG	No	0	Yes	Yes	Yes	No
ALST	Archive Last Value	DOUBLE	No	0	Yes	No	No	No
MLST	Monitor Last Value	DOUBLE	No	0	Yes	No	No	No
CLCV	Calculation String Valid	LONG	No	0	Yes	Yes	Yes	No

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
LA	Previous Input	DOUBLE	No	0	Yes	Yes	No	No
LB	Previous Input	DOUBLE	No	0	Yes	Yes	No	No
LC	Previous Input	DOUBLE	No	0	Yes	Yes	No	No
LD	Previous Input	DOUBLE	No	0	Yes	Yes	No	No
LE	Previous Input	DOUBLE	No	0	Yes	Yes	No	No
LF	Previous Input	DOUBLE	No	0	Yes	Yes	No	No
LG	Previous Input	DOUBLE	No	0	Yes	Yes	No	No
LH	Previous Input	DOUBLE	No	0	Yes	Yes	No	No
LI	Previous Input	DOUBLE	No	0	Yes	Yes	No	No
LJ	Previous Input	DOUBLE	No	0	Yes	Yes	No	No
LK	Previous Input	DOUBLE	No	0	Yes	Yes	No	No
LL	Previous Input	DOUBLE	No	0	Yes	Yes	No	No

Chapter 39:Waveform

1. Introduction

The waveform record type is used to interface waveform digitizers. The record stores its data in arrays. The array can contain any of the supported data types. The waveform's fields fall into the following categories:

scan parameters

read parameters

operator display parameters

run-time parameters

2. Scan Parameters

The waveform record has the standard fields for specifying under what circumstances the record will be processed. These fields are listed in *Scan Fields, Chapter 2, 2*. In addition, *Scanning Specification, Chapter 1, 1*, explains how these fields are used. Note that I/O event scanning is only supported for those card types that interrupt.

3. Read Parameters

These fields are configurable by the user to specify how and from where the record reads its data. How the INP field is configured determines where the waveform gets its input. It can be a hardware address, a channel access or database link, or a constant. Only in records that use soft device support can the INP field be a channel access link, a database link, or a constant. Otherwise, the INP field must be a hardware address. See *Address Specification, Chapter 1, 2*, for information on the format of hardware addresses and database links.

The DTYP field must contain the name of the appropriate device support module. The user can see a list of the device support modules currently supported at the user's local site by using the `dbst` utility (R3.13).

The values retrieved from the input link are placed in an array referenced by VAL. (If the INP link is a constant, elements can be placed in the array via dbPuts.) NELM specifies the number of elements that the array will hold, while FTVL specifies the data type of the elements.

The RARM field causes the device to re-arm when this field is set to 1.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
INP	Input Link	INLINK	Yes	0	No	No	N/A	No
NELM	Number of Elements in array	ULONG	Yes	1	Yes	No	No	No
FTVL	Field Type of Value	GBLCHOICE	Yes	0	Yes	No	No	No
RARM	Rearm	SHORT	Yes	0	Yes	Yes	No	Yes

4. Operator Display Parameters

These parameters are used to present meaningful data to the operator. They display the value and other parameters of the waveform either textually or graphically.

EGU is a string of up to 16 characters describing the units that the waveform measures. It is retrieved by the `get_units` record support routine.

The HOPR and LOPR fields set the upper and lower display limits for array elements referenced by the VAL field. Both the `get_graphic_double` and `get_control_double` record support routines retrieve these fields.

The PREC field determines the floating point precision with which to display the array values. It is used whenever the `get_precision` record support routine is called.

See Chapter 2, *Fields Common to All Record Types*, for more on the record name (NAME) and description (DESC) fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	null	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	0	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

5. Alarm Parameters

The waveform record has the alarm parameters common to all record types. *Alarm Fields, Chapter 2, 3*, lists other fields related to a alarms that are common to all record types.

6. Run-time Parameters

These parameters are used by the run-time code for processing the waveform. They are not configured using a configuration tool. Only the VAL field is modifiable at run-time.

VAL references the array where the waveform stores its data. The BPTR field holds the address of the array.

The NORD field holds a counter of the number of elements that have been read into the array. It is reset to 0 when the device is rearmed. The BUSY field indicates if the device is armed but has not yet been digitized.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
VAL	Value Field	See FTVL	No	0	Yes	Yes	Yes	Yes
BPTR	Buffer Pointer	NOACCESS	No	0	No	No	No	No
NORD	Number of Elements Read	ULONG	No	0	Yes	No	No	No
BUSY	Busy	SHORT	No	0	Yes	No	No	No

The following fields are used to operate the waveform in the simulation mode. See *Simulation Mode, Chapter 3, 2.4*, for more information on the simulation mode fields.

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
SIOL	Simulation Value Location	INLINK	Yes	0	No	No	N/A	No
SIML	Simulation Mode Location	INLINK	Yes	0	No	No	N/A	No
SIMM	Simulation Mode	GBLCHOICE	No	0	Yes	Yes	No	No
SIMS	Simulation Mode Alarm Severity	GBLCHOICE	Yes	0	Yes	Yes	No	No

7. Record Support Routines

init_record

Using NELM and FTVL space for the array is allocated. The array address is stored in the record.

This routine initializes SIMM with the value of SIML if SIML type is CONSTANT link or creates a channel access link if SIML type is PV_LINK. VAL is likewise initialized if SIOL is CONSTANT or PV_LINK.

This routine next checks to see that device support is available and a device support read routine is defined. If either does not exist, an error message is issued and processing is terminated

If device support includes init_record, it is called.

process

See next section.

get_value

Fills in the values of struct valueDes so that they refer to the array.

cvt_dbaddr

This is called by dbNameToAddr. It makes the dbAddr structure refer to the actual buffer holding the result.

get_array_info

Obtains values from the array referenced by VAL.

put_array_info

Writes values into the array referenced by VAL.

get_units

Retrieves EGU.

get_prec

Retrieves PREC if field is VAL field. Otherwise, calls `recGblGetPrec()`.

get_graphic_double

Sets the upper display and lower display limits for a field. If the field is VAL the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_control_double

Sets the upper control and the lower control limits for a field. If the field is VAL the limits are set to HOPR and LOPR, else if the field has upper and lower limits defined they will be used, else the upper and lower maximum values for the field type will be used.

get_graphic_double

Sets the following values:

upper_disp_limit = HOPR

lower_disp_limit = LOPR

get_control_double

Sets the following values

upper_ctrl_limit = HOPR

lower_ctrl_limit = LOPR

8. Record Processing

Routine process implements the following algorithm:

1. Check to see that the appropriate device support module exists. If it doesn't, an error message is issued and processing is terminated with the PACT field still set to TRUE. This ensures that processes will no longer be called for this record. Thus error storms will not occur.
2. Call device support read routine.
3. If PACT has been changed to TRUE, the device support read routine has started but has not completed writing the new value. In this case, the processing routine merely returns, leaving PACT TRUE.
4. Check to see if monitors should be invoked.
    - Alarm monitors are invoked if the alarm status or severity has changed.
    - Archive and value change monitors are always invoked.
    - NSEV and NSTA are reset to 0.
5. Scan forward link if necessary, set PACT FALSE, and return.

9. Device Support

9.1. Fields Of Interest To Device Support

Each waveform record must have an associated set of device support routines. The primary responsibility of the device support routines is to obtain a new array value whenever read_wf is called. The device support routines are primarily interested in the following fields:

Name	Summary	Description
PACT	Processing Active	See Chapter 2, *Fields Common to All Record Types* for an explanation of these fields.
DPVT	Device Private	
NSEV	New Alarm Severity	
NSTA	New Alarm Status	
INP	Input Link	This field is used by the device support routines to locate its input.

Name	Summary	Description
RATE	Sampling Rate	Some device support modules may find this useful.
PTSS	Pre-trigger Samples	Some device support modules may find this useful.
NELM	Number Of Elements In Array	
FTVL	Field Type Of Value	This is DBF_STRING, ... , DBF_ENUM. The device support routine should check that this is correctly defined.
RARM	Rearm	When set to 1, the device will be rearmed. The device support routine should reset it to 0 when done.
BPTR	Holds Address of Array	
NORD	Number of Elements Read	Device support must set this value when it completes.
BUSY	Is device busy?	

9.2. Device Support Routines

Device support consists of the following routines:

report

```
report(FILE fp, paddr)
```

Not currently used.

init

```
init()
```

This routine is called once during IOC initialization.

init_record

```
init_record(precord)
```

This routine is optional. If provided, it is called by the record support init_record routine.

get_ioint_info

```
get_ioint_info(int cmd,struct dbCommon *precord,IOSCANPVT
    *ppvt)
```

This routine is called by the ioEventScan system each time the record is added or deleted from an I/O event scan list. cmd has the value (0,1) if the record is being (added to, deleted from) an I/O event list. It must be provided for any device type that can use the ioEvent scanner.

read_wf

```
read_wf(precord)
```

This routine must provide a new input value. It returns the following values:

0: Success.

Other: Error.

9.3. Device Support For Soft Records

The `Soft Channel` device support module is provided to read values from other records and store them in arrays. If INP is a constant link, then read_wf does nothing. In this case, the record can be used to hold arrays written via dbPuts. If INP is a database or channel access link, the new array value is read from the link. NORD is set.

This module places a value directly in VAL.

If the INP link type is constant, then NORD is set to zero. If the INP link type is PV_LINK, then dbCaAddInlink is called by init_record.

read_wf calls recGblGetLinkValue which performs the following steps:

If the INP link type is CONSTANT recGblGetLinkValue does nothing.

If the INP link type is DB_LINK, then dbGetLink is called to obtain a new input value. If dbGetLink returns an error, a LINK_ALARM with a severity of INVALID_ALARM is raised.

If the INP link type is CA_LINK, then dbCaGetLink is called to obtain a new input value. If dbCaGetLink returns an error, a LINK_ALARM with a severity of INVALID_ALARM is raised.

NORD is set to the number of values returned and read_wf returns.

Appendix A: Menu Choices

1. GBLCHOICE and RECCHOICE Fields

Some of the fields in EPICS records are of type GBLCHOICE or RECCHOICE, meaning that the value of the field must be one from a menu of specific choices. There are several different menus available and each GBLCHOICE or RECCHOICE field uses one. For instance, the simplest menu is called `menuYesNo` and has two possible choices, NO and YES; no other values for a field that uses `menuYesNo` are possible—0.5, MAYBE, and NO WAY are not possible choices for the field. In addition, when a menu field is configured to have a certain value, its value string must match exactly one of the possible choices in its menu, including its case. For instance, if a field's type is GBLCHOICE and its menu is `menuYesNo`, the choices `yes`, `No`, and `Yes` are not valid choices: they don't match the case of NO or YES.

There are several menus provided as part of the standard EPICS release. Here are their names:

seqSELM
selSELM
menuYesNo
menuScan
menuPriority
menuOmsl
menuLinr
menuIvoa
menuFtype
menuConvert
menuCompress
menuArrType
menuAlarmStat
menuAlarmSevr
fanoutSELM
compressALG
aoOIF

A few of these menus are record-specific. For example, `aoOIF` is used solely by the OIF field in the analog output record. Any field using one of these menus will be of type RECCHOICE. Most of the above menus, however, are used by many different fields in different records. These are of type GBLCHOICE. For example, the `menuAlarmSevr` menu is used by the LLSV field in the analog input record as well as the COSV field in the binary input record, among others.

The above menus and their choices are provided as part of the standard EPICS release; however, developers can add to, delete, or change the choices of each, as well as add new menu types. (See the *Application Developer's Guide* for R3.13 for more information on how to add and change existing menus). Thus, the database designer should know that the available menus and their choices may vary from site to site. The designer must be familiar with the menu and menu choices for any fields he/she is configuring. This isn't a problem for most database designers because such tools as GDCT and Capfast allow the designer configuring the database to choose from the specified choices. Nonetheless, there are ways a designer can figure out the available choices for a particular menu and thus for a particular field without using a database configuration tool.

1.1. Determining the Choices of a GBLCHOICE or RECCHOICE Field

Using dbst

`dbst` is a utility that allows a user to run checks on a static database, a database that is not loaded and running. For instance, `dbst` can be used to make sure that all the field values in the database are valid. In addition, `dbst` can be used to view the definitions of record types, fields, and menus. To view the definition of a record, a field, or a menu, one doesn't need an actual database that has any record instances.

To view menu definitions, create a text file with the following lines:

```
path "/EPICSR3.13.0/base/dbd"
include "menuGlobal.dbd"
include "dbCommonRecord.dbd"
include "base.dbd"
```

You can call the file anything; however, it's better to append the extension `.db` to the file's name so that it's recognizable as a database file. The lines in this file use commands that are recognized by many EPICS database utilities, including `dbst`. The first line uses the `path` command to specify the directory that contains the database definition or `.dbd` files. The `.dbd` files contain definitions for record types, fields, menus, and breakpoints. The `.dbd` files are in a subdirectory of the `base` directory in the EPICS tree, wherever that is on your system. Thus, if the variable `$EPICS` points to the top level of the EPICS tree, the path `$EPICS/base/dbd` should be included in the quotes for the path command. `dbst` will use this directory to look for any `.dbd` files specified as part of the `include` command. The next three lines use the `include` command to include three `.dbd` files. The first `.dbd` file, `menuGlobal.dbd`, includes the definitions for all defined menus (actually, `menuGlobal.dbd` just includes the various `.dbd` files that actually contain the menu definitions); the second file, `dbCommonRecord.dbd`, defines all the fields common to all record types; and the third file, `base.dbd`, includes the `.dbd` files for all supported record types.

Specify the `.db` file you created on the command line when you start `dbst`. `dbst` is located in the extensions part of the EPICS tree for Release 3.13. For instance, if `$EPICS` points to the top level of the EPICS tree and your architecture is `sun4`, the following command would set your path so that you can run `dbst`:

```
set path = ($EPICS/extensions/bin/sun4 $path)
```

Then, by invoking `dbst` along with the name of the `.db` file you created, `dbst` will load the appropriate record type and field definitions, as well as the menu definitions. Although any of the options and arguments available with `dbst` can be specified from the command line, it's easier to simply specify the `-i` option which will cause `dbst` to enter the interactive mode. Thus, the following command runs `dbst`, causing it to load the database file as well causing it to enter the interactive mode:

```
dbst mydbfile.db -i
```

After `dbst` has read the appropriate database definition files, it will print the message `dbReadDatabase Completed` and you can enter in any commands.

To check the available choices for a field of type GBLCHOICE or RECCHOICE, you must know the name of the menu it uses if you don't already. To see the menu name associated with the field, use the command `-df`, specifying the record type and the field whose definition you want to see. In `dbst`, all commands and their arguments cannot contain spaces, the arguments being separated by commas instead (the -h or −? option will print out the available commands and their syntax). For instance, to check the definition for the OIF field from the Analog Output record, invoke the −df command as follows:

```
-df,ao,OIF
```

which will print the following field definition:

```
recordtype(ao)
    OIF
                 prompt: Out Full/Incremental
                  extra: (null)
          indRecordType: 49
                special: 0
             field_type: DBF_MENU
        process_passive: 0
                   base: 0
            promptgroup: GUI_OUTPUT
               interest: 1
               as_level: 1
                initial: (null)
                   menu: aoOIF
                   size: 0
                 offset: 0
```

As you can see in the third-to-the-last line, the name of the menu used by the OIF field is called aoOIF. The definition for the aoOIF menu can be viewed using either the −dm or the −wm commands and specifying the name of the menu as an argument.

```
-wm,aoOIF
```

prints the following definition:

```
menu(aoOIF) {
        choice(aoOIF_Full,"Full")
        choice(aoOIF_Incremental,"Incremental")
}
```

The choices for the field are specified in quotes—thus, the choices for the aoOIF menu are Full and Incremental and any other choices or mismatches like FULL, Increment, or Half-way are invalid. All menu definitions currently defined can be printed to standard output by invoking either of the commands −dm or −wm and not specifying a particular menu name.

Peeking at the Database Definition Files

The definition for a particular menu can also be seen simply by looking at the database definition or .dbd file for that menu. For instance, the menuAlarmSevr.dbd file contains the definition for the menuAlarmSevr menu. In addition, you can see which menu a GBLCHOICE field uses by looking at the dbCommonRecord.dbd file if the field is common to all record types or by looking at the file that defines that record type if the field is not command to all record types. For example, to see which menu the OIF field uses, one would have to look in the aoRecord.dbd file which contains the definition for the analog output record, because OIF is particular to the analog output record and is not common to all other record types. RECCHOICE menus are defined in the definition file for the record type in which they are used, usually at the beginning. For example, the field OIF is of type RECCHOICE and uses the aoOIF menu which is defined at the beginning of aoRecord.dbd.

2. Standard Menu Definitions

Here are the definitions for the menus available as part of the standard EPICS release. The designer should know that the definitions of these menus may be different at his/her site and these are provided merely as a convenience. It's recommended that you use the `dbst` utility to view menu choices. Remember that the actual choices appear in quotes.

seqSELM

```
choice(seqSELM_All,"All")
choice(seqSELM_Specified,"Specified")
choice(seqSELM_Mask,"Mask")
```

selSELM

```
choice(selSELM_Specified,"Specified")
choice(selSELM_High_Signal,"High Signal")
choice(selSELM_Low_Signal,"Low Signal")
choice(selSELM_Median_Signal,"Median Signal")
```

menuYesNo

```
choice(menuYesNoNO,"NO")
choice(menuYesNoYES,"YES")
```

menuScan

```
choice(menuScanPassive,"Passive")
choice(menuScanEvent,"Event")
choice(menuScanI_O_Intr,"I/O Intr")
choice(menuScan10_second,"10 second")
choice(menuScan5_second,"5 second")
choice(menuScan2_second,"2 second")
choice(menuScan1_second,"1 second")
choice(menuScan_5_second,".5 second")
choice(menuScan_2_second,".2 second")
choice(menuScan_1_second,".1 second")
```

menuPriority

```
choice(menuPriorityLOW,"LOW")
choice(menuPriorityMEDIUM,"MEDIUM")
choice(menuPriorityHIGH,"HIGH")
```

menuOmsl

```
choice(menuOmslsupervisory,"supervisory")
choice(menuOmslclosed_loop,"closed_loop")
```

menuLinr

```
choice(menuLinrNO_CONVERSION,"NO CONVERSION")
choice(menuLinrLINEAR,"LINEAR")
```

menuIvoa

```
choice(menuIvoaContinue_normally,"Continue normally")
choice(menuIvoaDon_t_drive_outputs,"Don't drive outputs")
choice(menuIvoaSet_output_to_IVOV,"Set output to IVOV")
```

menuFtype

```
choice(menuFtypeSTRING,"STRING")
choice(menuFtypeCHAR,"CHAR")
choice(menuFtypeUCHAR,"UCHAR")
choice(menuFtypeSHORT,"SHORT")
choice(menuFtypeUSHORT,"USHORT")
choice(menuFtypeLONG,"LONG")
choice(menuFtypeULONG,"ULONG")
choice(menuFtypeFLOAT,"FLOAT")
choice(menuFtypeDOUBLE,"DOUBLE")
choice(menuFtypeENUM,"ENUM")
```

menuConvert

```
choice(menuConvertNO_CONVERSION,"NO CONVERSION")
choice(menuConvertLINEAR,"LINEAR")
choice(menuConverttypeKdegF,"typeKdegF")
choice(menuConverttypeKdegC,"typeKdegC")
choice(menuConverttypeJdegF,"typeJdegF")
choice(menuConverttypeJdegC,"typeJdegC")
choice(menuConverttypeEdegF,"typeEdegF(ixe only)")
choice(menuConverttypeEdegC,"typeEdegC(ixe only)")
choice(menuConverttypeTdegF,"typeTdegF")
choice(menuConverttypeTdegC,"typeTdegC")
choice(menuConverttypeRdegF,"typeRdegF")
choice(menuConverttypeRdegC,"typeRdegC")
choice(menuConverttypeSdegF,"typeSdegF")
choice(menuConverttypeSdegC,"typeSdegC")
```

menuCompress

```
choice(menuCompressN_to_1_First_Value,"N to 1 First Value")
choice(menuCompressN_to_1_Low_Value,"N to 1 Low Value")
choice(menuCompressN_to_1_High_Value,"N to 1 High Value")
choice(menuCompressN_to_1_Average,"N to 1 Average")
```

menuArrType

```
choice(menuArrType8_bit_integers,"8 bit integers")
choice(menuArrType16_bit_integers,"16 bit integers")
choice(menuArrType32_bit_integers,"32 bit integers")
choice(menuArrTypeIEEE_floating_point,"IEEE floating point")
```

menuAlarmStat

```
choice(menuAlarmStat,"")
choice(menuAlarmStatREAD,"READ")
choice(menuAlarmStatWRITE,"WRITE")
choice(menuAlarmStatHIHI,"HIHI")
choice(menuAlarmStatHIGH,"HIGH")
choice(menuAlarmStatLOLO,"LOLO")
choice(menuAlarmStatLOW,"LOW")
choice(menuAlarmStatSTATE,"STATE")
choice(menuAlarmStatCOS,"COS")
choice(menuAlarmStatCOMM,"COMM")
choice(menuAlarmStatTIMEOUT,"TIMEOUT")
choice(menuAlarmStatHWLIMIT,"HWLIMIT")
choice(menuAlarmStatCALC,"CALC")
choice(menuAlarmStatSCAN,"SCAN")
choice(menuAlarmStatLINK,"LINK")
choice(menuAlarmStatSOFT,"SOFT")
choice(menuAlarmStatBAD_SUB,"BAD_SUB")
choice(menuAlarmStatUDF,"UDF")
choice(menuAlarmStatDISABLE,"DISABLE")
choice(menuAlarmStatSIMM,"SIMM")
choice(menuAlarmStatREAD_ACCESS,"READ_ACCESS")
choice(menuAlarmStatWRITE_ACCESS,"WRITE_ACCESS")
```

menuAlarmSevr

```
choice(menuAlarmSevrNO_ALARM,"NO_ALARM")
choice(menuAlarmSevrMINOR,"MINOR")
choice(menuAlarmSevrMAJOR,"MAJOR")
choice(menuAlarmSevrINVALID,"INVALID")
```

fanoutSELM

```
choice(fanoutSELM_All,"All")
choice(fanoutSELM_Specified,"Specified")
choice(fanoutSELM_Mask,"Mask")
```

compressALG

```
choice(compressALG_N_to_1_Low_Value,"N to 1 Low Value")
choice(compressALG_N_to_1_High_Value,"N to 1 High Value")
choice(compressALG_N_to_1_Average,"N to 1 Average")
```

```
choice(compressALG_Average,"Average")
choice(compressALG_Circular_Buffer,"Circular Buffer")
```

aoOIF

```
choice(aoOIF_Full,"Full")
choice(aoOIF_Incremental,"Incremental")
```