Motif 対応 XRT/graph™

プログラマーズ・ガイド&リファレンス・マニュアル

UNIX Edition ■ Version 4.0



株式会社タンジェント・コンピューティング
〒150-0001 東京都渋谷区神宮前 6-6-11
tel:(03)3409-5364
fax: (03)3409-0147
info@tan.co.jp
http://www.tan.co.jp

1999 年 5 月

本書は著作権法により保護されています。(株)タンジェント・コンピューティングによる事前の許可なく、本書のいかなる部分も無断転載、複製、複写を禁じます。
Copyright©1999 (株)タンジェント・コンピューティング

XRT、XRT/3d、XRT/care、XRT/field、XRT/gear、XRT/graph、XRT/table、XRT PDSはKL Group Inc.の登録商標です。
その他製品名、名称、サービス等は該当する会社または組織の商標または登録商標です。

XRT製品ライセンスについて

この度はXRT製品をご購入いただきまして誠にありがとうございます。

本製品のご利用に際しましては、次ページに記載の「XRTバイナリ使用許諾契約書」をご一読くださるようお願いいたします。下記に掲げる事項は、この契約の理解を補助するもので、契約の一部ではありません。

- XRT 製品(XRT/graph、XRT/3d、XRT/table 等)のライセンスは、機種(SUN SPARC、IBM RS/6000、HP 9000 等)につき、またツールキット(Motif)につき許諾されます。
- XRT 製品のライセンスにはシングルとネットワークの二種類があります。
- シングル・ライセンスでは、一つの CPUID で識別されるマシンにおいてエンドユーザ・アプリケーションの開発に XRT 製品をご使用いただくことができます。XRT 製品を使用する開発には下記の内容が含まれます。
  - XRT 製品のライブラリをエンドユーザ・アプリケーションのバイナリにコンパイル、リンクする。
  - XRT 製品のビルダーを実行する。
  - XRT 製品を GUI ビルダーまたは UIMS ツールに統合する。XRT 製品が統合されているツールは XRT 製品が使用許諾されているマシンでのみ使用できます。
- ネットワーク・ライセンスでは、ネットワークに接続されている同ビル内の同機種のマシンにおいてエンドユーザ・アプリケーションの開発に XRT 製品をご使用いただくことができます。
- 静的にリンクされている XRT 製品のライブラリを含むエンドユーザ・アプリケーションのバイナリは、ロイヤリティーやランタイム費用なしで提供していただくことができます。

XRTバイナリ使用許諾契約書

重要: ソフトウェア・パッケージを開封する前、またはソフトウェアをインストール、使用する前に必ずお読みください。

XRT製品の開封、インストール、使用、またはいずれかの左記の行為により、以下の規定につきお客様が承認されたものとさせていただきます。本契約書はお客様(個人または法人)とKL Group Inc.(以下、「KL Group」という。)の間で締結される法的な契約書です。もし、以下の規定に御同意頂けない場合には、ダウンロードやインストールされたファイルを直ちに破棄し、開封前のパッケージをライセンス費用の御支払いを示す書類と一緒に御返却ください。お客様がお支払になった料金は返却されます。

ソフトウェア・ライセンス

ライセンスの許諾 この使用許諾契約(以下、「ライセンス」という。)は、御指定のCPUと一致する一台のコンピュータにおいて、お客様が御購入されたXRTソフトウェア(以下、「ソフトウェア」という。)を御使用いただく権利をお客様に許諾するものです。ネットワーク・ライセンスを御購入された場合は、御指定のCPUと一致するコンピュータとローカル・エリア・ネットワークで接続された、同機種及び同ビル内に設置されているコンピュータにおいて、御購入いただいたライセンスで許可される御用台数を上限として「ソフトウェア」を御使用いただくことができます。また、共有ライブラリを御購入された場合、「ソフトウェア」のユーザ数は、御購入されたライセンスで許可された数を超えることはできません。「ソフトウェア」は、テンポラリ・メモリ(RAM等)にロードされた時、パーマネント・メモリ(ハード・ディスク、CD-ROM、その他の記憶装置等)にインストールされた時、エンドユーザ・アプリケーションにリンクされた時、開発ツール(コンパイラ、リンカー、GUIビルダー等)によってアクセスされた時に「使用」されたとします。他のコンピュータへ「ソフトウェア」のファイルを配布する目的のみで、ネットワークのファイル・サーバにインストールされている場合は、「使用」されているとはみなされません。

アップデート 本許諾契約書は、KL Groupから提供される「ソフトウェア」の全てのアップグレード版に適用されます。

著作権 「ソフトウェア」とそれに関連するイメージ、写真、テキスト等は、KL Groupとその供給者が所有するもので、その著作権は、カナダと米国の著作権法及び国際条約の規定で保護されています。従って、「ソフトウェア」は、書籍や音楽録音等、著作権で保護される他の製品と同じように扱ってください。「ソフトウェア」は、バックアップ目的で複製する以外は、一切複製作業を行うことはできません。また、「ソフトウェア」に付属する関連資料も同様に扱ってください。

その他の制限 「ソフトウェア」は、貸与、リース、譲渡、リバース・エンジニア、逆コンパイル、逆アセンブル、パスワードの作成をすることはできません。但し、これらの制限が司法管轄区域によって許可されていない場合は、この限りではありません。

「ソフトウェア」で構築されたアプリケーションの配布 「ソフトウェア」の"lib"または"lib-shared"ディレクトリにある、オブジェクト・コードのアプリケーション・ファイルにリンク可能な形態のファイル(以下、「XRTライブラリ・ファイル」という。)は、次の条件に従う場合にのみ、ロイヤリティなしで複製、配布することができます。(1)再リンクが不可能な方法でソフトウェアにリンクされて「XRTライブラリ・ファイル」が配布される。(2)配布されるソフトウェアは、「ソフトウェア」と同等または類似した機能を持つ開発ツールではない。(3)配布するソフトウェアの販売活動に、許可なくKL Groupの社名、ロゴ、トレード・マーク等を使用しない。(4)配布するソフトウェアには、著作権に関する注意書きを明記する。(5)配布したお客様のソフトウェアに対して第三者からなされた損害賠償請求に基づく損害等については、KL Group及びその供給者は一切その責任を負わない。それ以外の場合は、「ソフトウェア」のヘッダー・ファイルや(APIのドキュメンテーションを含む)ドキュメンテーション類を配布することはいかなる方法であってもできません。

U.S. Government Restricted Rights 「ソフトウェア」とその関連資料は、RESTRICTED RIGHTSに基づいて提供されています。本契約書で許諾される「ソフトウェア」とその関連資料は、Federal Acquisition Regulations (FAR) Part 12と Defense FAR Supplement (DFARS) 252.227-7014 (JUN 1995)で定義される"Commercial computer software"です。FAR 12.212 とDFARS 252.227.7014に基づき、この「ソフトウェア」の使用、複写、開示は、本契約書の条項に従います。製造者: KL Group Inc., 260 King Street East, Toronto, Ontario, Canada, M5A 1K3.

保証規定

品質保証 KL Groupは、「ソフトウェア」購入日より60日間、「ソフトウェア」がその関連資料記載の仕様に従って実質的に動作することを保証します。但し、品質保証の期間の制限が司法管轄区域によって許可されていない場合は、適用法例の範囲内でのみ有効です。尚、本品質保証は、第三者に譲渡することはできません。KL Groupによる故意、重大な過失により生じた損害、または製品に対する責任は、KL Groupは除外いたしません。

補償 KL Groupがその判断に基づいて行う補償は、次の事項に限定されます。(1)御支払いになられた金額を払い戻す。(2)上記記載の「品質保証」を満たさない「ソフトウェア」に対して補修、交換する。但し、(1)、(2)いずれの対応も、「ソフトウェア」と共ににその購入を証する書類を返却された場合にのみ提供されます。また、かかる事態が事故、故意若しくは過失、誤用等、KL Groupの責に帰さない理由により生じた場合は、本品質保証は無効になります。尚、交換された「ソフトウェア」の保証期間は、その受領より30日か、または交換の対象になった「ソフトウェア」の保証期間の残りか、どちらか長い方が適用されます。

権利侵害に対する補償 「ソフトウェア」が、第三者の特許、業務上の秘密、著作権等の侵害の対象となった場合、またはKL Groupがそのように判断した場合は、KL Groupは、次の補償を行うことがあります。(1)「ソフトウェア」を継続して使用できる権利を取得する。(2)「ソフトウェア」を権利侵害の対象にならないように交換、修復する。但し、交換または修復された場合も、以前と同じ機能は保つようにする。上記(1)及び(2)が適正でない場合は、(3)「ソフトウェア」のライセンス許諾を中止し、削除する。その場合、ライセンスに対して支払われた金額の内、定額法を適用した5年間の償却期間に基づいた相当金額を返却する。

保証の制限 司法管轄区域の適用法令の範囲内で、KL Group及びその供給者は、「ソフトウェア」とその関連資料の商品性、特定目的適合性の保証を含め、本保証規定に規定されていないその他の保証を、明示たると黙示たるとを問わず一切いたしません。この有限保証は、お客様に特定な法的権利を与えるものですが、州、管轄区域によっては、その他の権利が与えられることもあります。。

免責 司法管轄区域の適用法令の範囲内で、KL Group及びその供給者は、「ソフトウェア」の使用または使用不能から生ずる損害(事業利益の損失、事業の中断、事業情報の損失または金銭的損害を含むがこれらに限定されない)に関して、一切責任を負わないものとします。例え、KL Groupがかかる損害の可能性について知らされていた場合も同様です。但し、州や管轄区域によっては、必然的または偶発的な損害に対する責任の除外や制限を認めていない場合があり、その場合、前述の制限は適応されないことがあります。

適用法令 お客様が"European Union"に属する地域に居住されている場合、本契約書はその国の法律に準拠し、契約当事者双方は、その国の司法に従います。それ以外の場合は、本契約書は、カナダ・オンタリオ州の法律に準拠し、契約当事者双方は、オンタリオ集の司法に従います。

パブリック・ドメイン・ソフトウェア 本契約書に記載されている条項は、パブリック・ドメインと明記されているソフトウェアには適用されません。

本契約書は、KL Groupより提供される"XRT FOR MOTIF BINARY LICENSE AGREEMENT"を(株)タンジェント・コンピューティングが日本語訳したものです。表現等に不一致がある場合は、"XRT FOR MOTIF BINARY LICENSE AGREEMENT"次ページ参照)に従います。

XRT FOR MOTIF BINARY LICENSE AGREEMENT

Important — Read Carefully Before Opening Software Package(s), Installing or Using Software On Your Computer

By opening a sealed software package, or installing, or using this XRT product, you indicate your acceptance of the following XRT for Motif Binary License Agreement. This is a legal agreement between you (either an individual or an entity) and KL Group Inc. If you do not accept and agree with all of the terms of this agreement, promptly destroy any downloaded or installed files, and return the unopened software package(s) and all other materials with proof of payment to your place of purchase, and your license fee will be refunded.

SOFTWARE LICENSE

Please Note: This is a license agreement and not an agreement for sale.

Grant of License. This License Agreement ("License") permits you to use one copy of the software you purchased (the "SOFTWARE"), on a single computer after registration of that computer's CPU identification with us. If you have purchased a network site license you may also use the SOFTWARE on computers in the same building having the same architecture as a registered computer, and connected by the same local area network, provided that the total number of such computers, on which the SOFTWARE is ever used, does not exceed the maximum number permitted by the network site license which you ordered.

If you have purchased shared library versions of the SOFTWARE, the number of users of the SOFTWARE cannot exceed the maximum number permitted by the license which you ordered.

The SOFTWARE is "in use" on a computer when it is loaded into temporary memory (i.e. RAM); when it is installed into permanent memory (e.g. hard disk, CD-ROM, or other storage device); when it is being linked into an end-user application; or when it is accessed from a software development tool such as a compiler, a linker, or a GUI builder. The SOFTWARE is not considered "in use" when installed on a network file server for the sole purpose of file distribution to other computers.

Updates. This agreement will govern the terms of any SOFTWARE updates provided to you by KL Group.

Copyright. The SOFTWARE (including any images, photographs and text incorporated into the SOFTWARE) is owned by KL Group or its suppliers and is protected by Canadian and United States copyright laws and international treaty provisions. Therefore you must treat the SOFTWARE like any other copyrighted material (e.g., a book or musical recording) and not reproduce it except that you may make a backup for archival purposes. You may not copy the printed materials accompanying the SOFTWARE.

Other Restrictions. You may not rent, lease, transfer, reverse engineer, decompile, disassemble, create passwords for or translate the Software, except to the extent such foregoing restriction is expressly prohibited by applicable law.

Distribution of applications you build with the SOFTWARE. You have a royalty-free right to reproduce and distribute files in the SOFTWARE that are found in the "lib" or "lib-shared" directories and which are in a form suitable for linking into object code application files (the "XRT Library Files") provided that (a) you distribute the XRT Library Files only when linked into your software product in such a way as to prohibit further relinking; (b) your software product is not an application development tool that provides similar functionality to the SOFTWARE licensed hereunder; (c) you do not use KL Group's name, logo or trademark to market your software product; (d) you include a valid copyright notice on your software product; and (e) you agree to indemnify, hold harmless, and defend KL Group and its suppliers from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of your software product. Unless otherwise specified, you may not in any manner distribute any header files or documentation (including API documentation) provided as part of the SOFTWARE.

U.S. Government Restricted Rights. The SOFTWARE and documentation are provided with RESTRICTED RIGHTS. The SOFTWARE (including documentation) licensed in this agreement is "Commercial computer software" as defined in Federal Acquisition Regulations (FAR) Part 12 and Defense FAR Supplement (DFARS) 252.227-7014 (JUN 1995). In accordance with FAR 12.212 and DFARS 252.227.7014, notwithstanding any provision to the contrary, use, duplication and disclosure of this software is governed by the terms of this agreement. Manufacturer is KL Group Inc., 260 King Street East, Toronto, Ontario, Canada, M5A 1K3.

LIMITED WARRANTY

Limited Warranty. KL Group warrants that the SOFTWARE will perform substantially in accordance with the accompanying written materials for a period of sixty (60) days from the date of receipt. Some states/jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you. This warranty may not be assigned. KL Group does not exclude product liability or liability for damages resulting out of intent or gross negligence on the part of KL Group or its leading personnel.

Customer Remedies. KL Group's and its suppliers' entire liability and your exclusive remedy shall be, at KL Group's option, either (a) return of the price paid, or (b) repair or replacement of SOFTWARE that does not meet KL Group's Limited Warranty and which is returned to KL Group with a copy of your receipt. This Limited Warranty is void if failure of the SOFTWARE has resulted from accident, abuse, or misapplication. Any replacement SOFTWARE will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer.

Customer Remedies for Infringement. Should the SOFTWARE become or in KL Group's opinion be likely to become the subject of a claim for infringement of patent, trade secret or copyright, KL Group may (a) procure for you the right to continue to use the SOFTWARE; or (b) replace or modify the SOFTWARE to make it non-infringing, provided that the same function is performed by the replacement or modified SOFTWARE; or if (a) or (b) are not commercially reasonable, (c) terminate the license to use the SOFTWARE, remove the SOFTWARE, and grant you a credit for the license fee as depreciated on a straight-line five (5) year basis.

No Other Warranties or Obligations.. To the maximum extent permitted by applicable law, KL Group and its suppliers shall have no other obligations and disclaim all other warranties, either express or implied, including, but not limited to, implied warranties of merchantability and fitness for a particular purpose, with regard to the SOFTWARE and the accompanying printed materials. This limited warranty gives you specific legal rights. You may have others which vary from state/jurisdiction to state/jurisdiction.

No Liability for Consequential Damages. To the maximum extent permitted by applicable laws, in no event shall KL Group or its suppliers be liable for any damages whatsoever (including without limitation, damages for loss of business profits, business interruption, loss of information, or other pecuniary loss) arising out of the use of or inability to use this KL Group product, even if KL Group has been advised of the possibility of such damages. Because some states/jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

Governing Law. If you are a resident of one of the member states of the European Union, this Agreement is governed by the laws of your country of residence, and each of the parties hereto irrevocably attorns to the jurisdiction of the courts of your country of residence. Otherwise, this Agreement is governed by the laws of the Province of Ontario, Canada. Each of the parties hereto irrevocably attorns to the jurisdiction of the courts of the Province of Ontario.

Public Domain Software. This license agreement and the warranties, remedies, limitations, conditions and restrictions contained herein do not apply to any software clearly marked as being in the public domain.

BINLIC-M-11/98

目次

はじめに ····· 1

XRT/graph について ····· 1
XRT/graph 4.0 の新しい機能 ····· 2
前提知識 ····· 3
表記上の規約 ····· 3
関連資料 ····· 3
テクニカル・サポート ····· 4
お客様のご意見 ····· 4

Part I XRT/graph の使用法

1 XRT/graph の 簡単なプログラムの開発 ····· 9

1.1 はじめに ····· 9
1.2 環境変数 ····· 9
1.3 コンパイルとプログラムの許可 ····· 10
1.4 基本的な線グラフ ····· 11
1.5 データのロード ····· 13
1.6 ヘッダー、フッター、ラベル ····· 13
1.7 棒グラフに変更する ····· 15
1.8 グラフのインバート ····· 16
1.9 グラフを完成させる ····· 17
1.10 サンプル・コード ····· 18

2 XRT/graph の基本操作 ····· 21

2.1 Motif との統合 ····· 21
2.2 用語の解説 ····· 21
2.3 グラフ・タイプ ····· 24
2.4 浮動小数点のリソース ····· 27
2.5 UseDefault リソース ····· 27
2.6 ポインタ・リソース ····· 28
2.7 ストリング・リソース ····· 28
2.8 色 ····· 29
2.9 リソースファイルの使用 ····· 31

3 プロパティ・エディタの使用 ・・・・・・・・・・・・・・・・・・・・・・・・ 33

3.1 はじめに・・・・・・・・・・・・・・・・・・・

4.19 スクロールバー……94
4.20 出力と印刷……95

5 XRT/graph Data …… 97

5.1 グラフへデータを取り込む……97
5.2 データ・オブジェクト……98
5.2.1 Array と General データのフォーマット ……99
5.2.2 XrtDataHandle の属性……100
5.3 オブジェクトとデータ・アクセス法……100
5.4 データの除外と非表示……103
5.5 データのトランスポーズ……104
5.6 複数のデータ・セット……105
5.7 静的データを使用した例……106
5.8 パイプからリアルタイム・データを読み込む例 ……107

6 ユーザ・インタラクションの プログラミング …… 109

6.1 デフォルトのユーザ・インタラクション……109
6.2 トランスレーションとアクションの概要……109
6.3 ユーザ・インタラクションの開始……111
6.4 ユーザ・インタラクションのアップデート……112
6.4.1 スケーリング、移動、ズーム……112
6.4.2 軸のズーム ……113
6.4.3 スケーリング、移動、ズームのリセット……114
6.4.4 回転 ……115
6.4.5 インタラクションの不許可……116
6.5 ユーザ・インタラクションの終了……116
6.6 グラフ・データとのインタラクション……117
6.6.1 アクション・ルーチンとコールバック……117
6.6.2 イベント・ハンドリングの手続き……119
6.7 ウィンドウのリサイズ……125

7 高度なプログラミング …… 127

7.1 二番目の Y 軸 ……127
7.2 複合グラフ……128
7.3 テキスト・オブジェクト……130
7.4 リソース・アップデートのバッチ処理……133
7.5 国際化サポート……133
7.6 高速アップデート機能……135
7.7 MS Windows へのポーティング ……136
7.8 警告メッセージのカスタマイズ……140
7.9 C++の使用……140

7.10 UIL の使用 ····· 141
7.11 リソース値をストリングに変換する ····· 141

8 PDS ユーティリテイ・ライブラリ ····· 143

8.1 XrtPDSImage ····· 143
8.1.1 XrtPDSImage の作成 ····· 143
8.2 Image 型の登録 ····· 144
8.2.1 定義済みのイメージ ····· 144
8.2.2 任意のイメージ ····· 145
8.2.3 イメージ・ファンクションの書き込み ····· 147
8.3 ダイナミックまたは共用ライブラリの使用 ····· 148
8.4 XrtPDSList オブジェクト ····· 148
8.4.1 サンプル・コード ····· 149
8.4.2 リストの作成と破棄 ····· 149
8.4.3 リストのアイテムの追加と削除 ····· 149
8.4.4 アイテムの検索と取得 ····· 151
8.4.5 リスト比較ルーチン ····· 154
8.4.6 リストのソート ····· 155
8.4.7 ユーザが定義したデータ ····· 156
8.5 手続きと方法 ····· 156
8.5.1 XrtPDSImage の方法 ····· 156
8.5.2 XrtPDSList オブジェクトの方法 ····· 158

Part II リファレンス

A リソースの一覧 ····· 171

A.1 ウィジェットの概要 ····· 171
A.2 XRT/graph のリソース ····· 171
A.3 継承されるリソース ····· 203

B テキスト・オブジェクトの属性 ····· 205

C PointStyle オブジェクトの属性 ····· 209

D 手続きと方法 ····· 211

E トランスレーションとアクション ····· 251

E.1 デフォルトのトランスレーション ····· 251
E.2 アクション・ルーチン ····· 252

F データ型 ····· 257

G リソース名、型とクラス ・・・・・・・・・・・・・・・・・・・・・・・・・・・ 267

H リソース・ファイルの使用 ・・・・・・・・・・・・・・・・・・・・・・・・・ 277

H.1 構文・・・・・・・・・・・・・・・・・・・

はじめに

XRT/graph について ■ XRT/graph 4.0 の新しい機能
前提知識 ■ 表記上の規約
関連資料 ■ テクニカル・サポート
お客様のご意見

XRT/graph について

XRT/graph はボタン、スライダ、メニュー等のウィジェットから構成されている Motif に新たなウィジェットを追加します。データはビジュアルに表示され、ユーザは、インタラクティブにグラフを操作することができます。

グラフ・ウィジェットは、グラフの表示や動作をコントロールするリソースを備えています。ウィジェットが一つ増えた点を除けば、XRT/graph を使用したプログラミングは他の Motif のプログラミングとほとんど違いはありません。

XRT/graph は、XRT/3d や XRT/table と組み合わせて使用することができます。XRT/3d は、3D データをサーフェス、等高線、棒グラフに表示し、XRT/table は、行と列の表形式のデータを表示します。

グラフ・ウィジェットには、以下の内容をコントロールするリソースがあります。

- グラフ・タイプ(線、面、棒、積上げ棒、円、バブル、Hilo、HiloOpenClose、キャンドル、ポーラー、レーダー、filled レーダー、複合グラフ)
- ヘッダー/フッターの位置、枠線、テキスト、フォント、色
- データ・スタイル(線や塗りつぶしの色とパターン、線幅、ポイントのスタイル/サイズ/色)
- 凡例の位置、方向、枠線、アンカー、フォント、色
- グラフの位置、枠線、色、幅、高さ、3D 効果
- 軸ラベル(ポイント・ラベル、セット・ラベル、値ラベル、時間ラベル)
- 軸とデータの最大・最小、数値ラベル/目盛りのインクレメント、グリッド線のインクレメント、フォント、原点、軸の方向、精度
- 軸、注釈、原点の位置、軸の反転

- コールバック・リソースを使用した、ユーザ・インタラクションのコントロール
- マーカとテキスト領域

XRT/graph には、次の作業を行う手続きや方法が提供されています。

- グラフ化される数値を含んだデータ・オブジェクトの割り当てやロード
- データ・オブジェクトに含まれたデータやエレメントのアップデート

瞬時のデータ表示

グラフの JPEG、PNG 出力

- プロパティ・エディタにより、実行時にリソースを変更する
- グラフの PostScript 出力
- インデックス付きリソースの設定、受け取り支援

XRT/graph 4.0 の新しい機能

新しいグラフ

- バブル・チャート
- 積み上げた形式の面や線グラフ、レーダ・チャート
- スライスが切り離された円グラフ

円グラフの開始点の指定

表示に関する改良点

各ポイントに対して設定できるデータ・スタイル

各軸に対して設定できるデータ・スタイル

タイトル付きの凡例

- 種類が増えたポイント・スタイルとデフォルトのデータ・スタイル
- グラフ、ヘッダー/フッター、テキスト領域の背景にはイメージが表示可能

PDS ユーティリティ・ライブラリ

PDS ユーティリティ・ライブラリを利用して、JPEG、PNG など、ユーザが定義したイメージ・フォーマットに出力できるようになりました。XrtPDSImage は、イメージを持つオブジェクトです。Pixmap は内部で使用されますが、この新しい方法によって、様々なイメージのフォーマットをサポートすることができます。

このライブラリは、その他、XrtPDSList オブジェクトもサポートしています。これを利用して、複数のアイテムのグループを簡単に記憶させることができます。

その他の改良点

新しく改良された例とデモ・プログラム

バグ修正とマイナー・アップグレード(修正されたバグについては、リリース・ノートを参照してください。

前提知識

このマニュアルは、C 言語及び Motif ツールキットに精通していることを前提として書かれています。Motif や XRT/graph アプリケーションをプログラミングするには、C 言語の"char *の配列"、"構造体のポインタ"等を理解している必要があります。またリソース値の取得や設定等の Motif の基本的な概念も理解していなければなりません。Motif に関する資料については 関連資料の項をご覧ください。

表記上の規約

`Typewriter Font`

- C コードやファイル内容の例
- XRT/graph や Motif リソース名、定数
- 端末で入力するコマンド

斜体

- パス、ファイル名、プログラム、手続き、パラメータ
- 新しい用語や重要な用語
- 図やテーブル
- このマニュアルで参照されている参考資料名(Motif Programming Manual 等)

`...[XYY2]...`

軸の情報を指定する XRT/graph のリソースは、X、Y、二番目の Y 軸、どれも同様です。リソース名の[XYY2]は、X 軸、Y 軸、二番目の Y 軸に対応するリソースを意味します。例えば "`XtNxrtXNum`" 、 "`XtNxrtYNum`" 、 "`XtNxrtY2Num`" は 、"`XtNxrt[XYY2]Num`"となります。

関連資料

Motif Programming Manual と *Motif Reference Manual*(O'Reilly & Associates,Inc 出版)

X Toolkit Intrinsics Reference Manual、*Xlib Programming Manual*、*Xlib Reference Manual*(O'Reilly & Associates,Inc.出版) これらのマニュアルは XRT/gear や Motif の開発には直接必要ありませんが、Xlib の知識は X Window アプリケーションの開発に重要です。

テクニカル･サポート

初期の問題の多くは、インストールやセットアップに関する問題です。リリースノート&インストールガイドには、インストールやセットアップの際の問題の回避法がありますので参照してください。

XRT テクニカル･サポートへお問い合わせされる場合は、次の事項をご連絡ください。

- XRT/field ライセンスのシリアル番号
- X と Motif のバージョン
- OS のバージョン

オンライン･サポート

(株)タンジェント･コンピューティング及び、開発元の KL Group Inc 社の web サイトには、技術的な情報が掲載されていますので参照してください。

技術情報及び FAQ **(日本語)**	http://www.tan.co.jp/xrt/support/index.htm
技術情報(英語)	http://www.klgroup.com/cs/tech/xrt/
Knowledge Base(**英語**)	http://www.klgroup.com/cgi-bin/xrt_search.cgi
FAQ(**英語**)	http://www.klgroup.com/cs/tech/xrt/faq/

尚、テクニカル･サポートを御受けいただくには、XRT/care と呼ばれる、保守サポート･サービスを御購入されていることが前提となります。XRT/care のお申し込み法やサービスに関する御質問等がございましたら、(株)タンジェント･コンピューティングまでご連絡ください。

お客様のご意見

XRT 製品のアップグレードで追加される新しい機能の大半は、お客様のリクエストに基づいています。XRT/graph に関するお客様のご意見、ご使用例、或は、あれば便利と思われる機能等がございましたら下記までご連絡ください。

株式会社タンジェント･コンピューティング
〒150-0001 東京都渋谷区神宮前 6-6-11
電話 03-3409-5364
Fax 03-3409-0147
Email info@tan.co.jp
WWW http://www.tan.co.jp/

Part I

Using XRT/graph

1

XRT/graph の 簡単なプログラムの開発

はじめに ■ 環境変数

コンパイルとプログラムの許可 ■ 基本的な線グラフ

データのロード ■ ヘッダー、フッター、ラベル

棒グラフに変更する ■ グラフのインバート

グラフを完成させる ■ サンプル・コード

1.1 はじめに

この章では、*plot1.c* というサンプルのプログラムを実際にコンパイル、実行しながら、XRT/graph の基本的なプログラミングを紹介します。このプログラムは架空の"Michelle's Microchips 社"の四半期ごとの収入、支出データをグラフ化します。

次の表はグラフ化されるデータです。

	Q1	Q2	Q3	Q4
支出	150.0	175.0	160.0	170.0
収入	125.0	100.0	225.0	300.0

1.2 環境変数

最初に UNIX の環境をご確認ください。*XRTHOME* という環境変数が、XRT/graph のファイルのディレクトリ名に定義されていなければなりません。

次のように入力して、XRTHOME の現在の値を確認できます。

```
echo $XRTHOME
```

もしXRTHOME環境変数が正しく設定されていない場合は、リリース&インストレーション・ガイドを参照するか、システム管理者の方にお問い合わせください。

また Motif のアプリケーションがコンパイルできることもご確認ください。ご使用になっている Motif のマニュアルで、Motif のライブラリとインクルード・ファイルの位置を調べることができます。XRT/graph を使用される前に、簡単な Motif のアプリケーションがコンパイルでき稼動することも必ずご確認ください。

1.3 コンパイルとプログラムの許可

もし、X、Motif、XRT/graph ライブラリ、インクルード・ファイルが標準の場所(*/usr/lib*、*/usr/include/X11*、*usr/include/Xm*)にインストールされていれば、*plot1.c* は次のようにしてコンパイルや許可を与える事ができます。[1]

```
cc plot1.c -I$XRTHOME/include \
  -L$XRTHOME/lib -o plot1 \
  -lxrtm -lpdsutil -lXm -lXpm -lXt -lX11 -lm
$XRTHOME/bin/xrt_auth plot1
```

もし、ライブラリとインクルード・ファイルが標準の位置になければ、`-I` と `-L` のコンパイラ・フラグが必要です。

xrt_auth

XRT/graph のライブラリが実行可能アプリケーションにリンクされたら、*xrt_auth* を使用して許可を与えます。*xrt_auth* は、ご使用のマシンに XRT/graph のライセンスがあるかを調べ、アプリケーションに許可を与えます。*xrt_auth* には、`-v`(verbose message;詳細メッセージ)と`-q`(quiet mode;メッセージ出力なし)の二つのフラグのオプションがあります。

```
xrt_auth [-q | -v] executable_file
```

エンドユーザ・アプリケーションは *xrt_auth* により許可が与えられると、どのマシンでも実行することができます。XRT/graph で作成されたアプリケーションにはランタイム費用はかかりません。

アプリケーションが *xrt_auth* によって許可が与えられないと、XRT/graph ウィジェットは数秒後に消滅し、図1のメッセージが現れます。

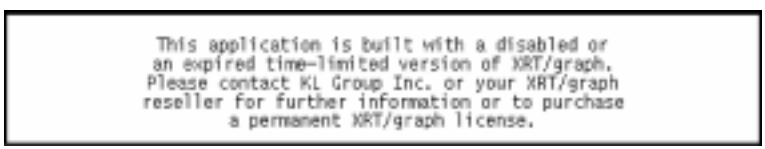


図1　期限切れまたは未許可の場合のメッセージ

[1] コンパイルのオプションは機種に依存します。御使用のマシンでのコンパイル例が、*$XRTHOME/src/graph/examples* の Makefile にありますので参照してください。

注意: 共用ライブラリのリンクや許可に関しては、リリースノートを参照してください。

C++と ANSI C

XRT/graph の include ファイルは、C++と ANSI C、"Kernighan and Ritchie"C との互換性があります。

1.4 基本的な線グラフ

図2 は *plot1.c* のプログラムです。このプログラムは、ファイルからデータをロードし、Motif のシェル･ウィジェットを作成、そしてデータをグラフ化するグラフ･ウィジェットを作成する最小限の C プログラムです。このマニュアルのほかのデモプログラムと同様に、このファイルは*$XRTHOME/src/graph/examples* ディレクトリにあります。

```
1 #include <X11/Intrinsic.h>
2 #include <Xm/Xm.h>
3 #include <Xm/XrtGraph.h>
4
5 int main(int argc, char *argv[])
6 {
7  XtAppContext app_context;
8  Widget graph, toplevel;
9  XrtDataHandle hData;
10
11 XtSetLanguageProc(NULL,NULL,NULL);
12
13 /* データを割り当てロードする */
14 hData = XrtDataCreateFromFile("data/mm63.dat", NULL);
15
16 /* Intrinsics を初期化する */
17 toplevel = XtVaAppInitialize(&app_context, "Plot 1", NULL, 0,
18     (Cardinal *) &argc, argv, NULL, NULL);
19
20 /* グラフの作成 */
21 graph = XtVaCreateManagedWidget("mygraph",
22     xtXrtGraphWidgetClass, toplevel,
23     XtNxrtData, hData,
24     NULL);
25
26 XtRealizeWidget(toplevel);
27 XtAppMainLoop(app_context);
28 }
```

図2　plot1.c

plot1.c のコードは、一般的な Motif プログラミングの形です。11 行目では、XtSetLanguageProc(NULL、NULL、NULL)が呼び出され、ロケールが初期化されています。14 行目は、(XrtDataHandleData hData で参照される)データ･オブジェクトを割り当て、*data/mm63.dat* ファイルのデータをオブジェクトにロードします。このプログラムが実行される時に、カレント･ディレクトリの下に *data* というディレクトリがあることを確認しておいてください。

17 行目では Intrinsics がイニシャライズされ、トップレベルのシェル・ウィジェットが標準の Xt Intrinsics 形式で作成されています。21 から 24 行目では、Xt Intrinsics の *XtVaCreateManagedWidget()*[1] が、`toplevel` の管理された子であるグラフ・ウィジェットを生成しています。グラフが作成されると、`XtNxrtData` リソースは `hData` にセットされます。

標準の Xt の手続き(*XtSetValues()*、*XtGetValues()*、*XtVaSetValues()*)がグラフ・ウィジェットの操作に使用できます。グラフ・ウィジェットは、Motif Primitive のサブクラスですので、グラフのリソースの他に Primitive と Core のリソースも使用することができます。(例えば、グラフ・ウィジェットに `XmNshadowThickness` を設定することができます。)

XRT/graph は 12 種類(type)のグラフ(線、面、棒、積上げ棒、円、HiLo、HiLoOpenClose, キャンドル、ポーラー、レーダー、filled(塗りつぶし)レーダー)をサポートします。グラフのタイプは `XtNxrtType` で指定します。デフォルトは線グラフなので、*plot1.c* の `XtNxrtType` リソースを指定する必要はありません。

plot1.c がコンパイル、実行されると図 3 が表示されます。

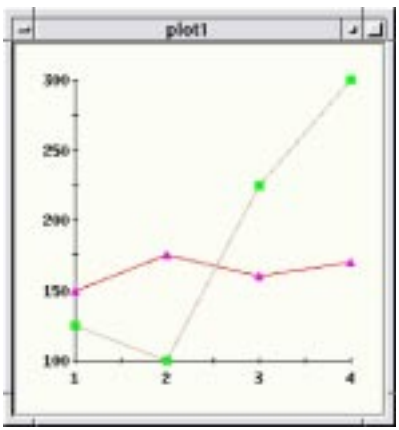


図 3　plot1 のウィンドウ

[1] このマニュアルでは Varargs インターフェースが使用されています。ウィジットの作成(*XtVaCreateManagedWidget()*、*XtVaCreateWidget()*)、リソースの設定と取得(*XtVaSetValues()*、*XtVaGetValues()*)の Varargs インターフェースがあります。詳しくは Xt Intrinsics のマニュアルを参照してください。

1.5 データのロード

XRT/graph でグラフを表示するには、グラフ・データ(*dataset*)を XRT/graph が使用できるフォーマットでロードする必要があります。ファイルからデータを割り当て、ロードするには *XrtDataCreateFromFile()*手続きを使用してください。[1]

図 4 は、*mm63.dat* というファイルの内容です。このファイルは、*XrtDataCreateFromFile()*が理解できるフォーマットです。'#'で始まる行は、コメントとして扱われます。1 行目は、このデータが XRT_DATA_ARRAY スタイルで、2 セット(*set*)、4 ポイント(*point*)で構成されていることを *XrtDataCreateFromFile()* に伝えます。XRT/graph は、XRT_DATA_ARRAY と XRT_DATA_GENERAL の二つのスタイルをサポートします。

```
ARRAY 2 4
# X-values
1.0 2.0 3.0 4.0
# Y-values set 1
150.0 175.0 160.0 170.0
# Y-values set 2
125.0 100.0 225.0 300.0
```

図4　mm63 ファイル

mm63.dat ファイルは、2 セット、4 ポイントを定義します。3 行目は、二つのセット(Y 値)で共有される X 値(ポイント)です。この場合、各 X 値は表 1 の各列(Q1、Q2、etc.)に対応します。5 行目と 7 行目は、Y 値の各セットを定義します。

XRT/graph のデータには、ARRAY(配列)と GENERAL(一般)の二つのデータ型があります。複数の Y 値のセットが共通の X 値を共有するときは、ARRAY を使用します。共通の X 値をシェアしない時や、全てのセットが同じ個数の値を持っていない時は GENERAL を使用します。

1.6 ヘッダー、フッター、ラベル

plot1.c はまだ未完成です。ヘッダー、フッター、凡例がありませんし、X 軸の数値も意味がありません。XRT/graph では、指定されるリソースが少なくても、デフォルト値を使用して完成度の高いグラフを表示します。[2]

グラフのリソースは、全て、グラフ・ウィジェットが作成される時に指定できます。またリソースは、*XtSetValues()*、*XtVaSetValues()*を用いて、いつでも変更することができます。

---

[1] より一般的な方法として、*XrtDataCreate()*を用いてデータ・オブジェクトを割り当て、次に、C でデータをオブジェクトにセットすることができます。

[2] XRT/graph によって計算されるデフォルトは変化します。例えば、X 軸の数値ラベルの増分を決定する XtNxrtXNum のデフォルトは、現在のグラフ領域のサイズ(ウィンドウのリサイズによって変化)や XtNxrtNumMethod、XtNxrtXPrecision の値によって、決定されます。

*XtGetValues()*または *XtVaGetValues()*を使用すると、リソースの現在の値を調べることができます。

ヘッダー、フッター、凡例、X 軸のラベルを作成するには、*XtVaCreateManagedWidget()*にリソースを幾つか付け加える必要があります。次の行は、グラフの上下にヘッダーとフッターを指定します。

```
XtNxrtHeaderStrings, header_strings,
XtNxrtFooterStrings, footer_strings,
```

どちらのリソースも NULL 終端文字配列を値として取ります。これらは、次の例のように定義してください。

```
static char *header_strings[] = {"Michelle's Microchips", NULL};
static char *footer_strings[] = {"1963 Quarterly Results", NULL};
```

デフォルトでは、ヘッダー、フッター、凡例のストリングは"fixed"フォントで表示されます。ヘッダーはグラフの上、フッターはグラフの下に、それぞれ中央揃えに配置され、凡例はグラフの右に配置されています。

XRT/graph で使用される全ての配列データは、"*m* 個のポイント(*point*)が *n* 個セット(*set*)ある"とみなされます。この例では 4 ポイントが 2 セットあります。各セットとポイントのラベルは以下のように指定できます。

```
static char *set_labels[] = {"Expenses",
  "Revenue", NULL};
static char *point_labels[] = {"Q1", "Q2",
  "Q3", "Q4", NULL};
```

XRT/graph は、自動的にセット・ラベルを凡例に使用します。X 軸にポイント・ラベルを使用するには、XtNxrtXAnnotationMethod を XRT_ANNO_POINT_LABELS に設定してください。グラフは、次のコールで作成されます。

```
graph = XtVaCreateManagedWidget("mygraph",
  xtXrtGraphWidgetClass,    toplevel,
  XtNxrtData,           my_data,
  XtNxrtHeaderStrings,      header_strings,
  XtNxrtFooterStrings,      footer_strings,
  XtNxrtSetLabels,      set_labels,
  XtNxrtPointLabels,        point_labels,
  XtNxrtXAnnotationMethod,  XRT_ANNO_POINT_LABELS,
  XtNwidth,             500,
  NULL);
```

図5は、上のコードで得られた結果です。

Core のリソース、XtNwidth が 500 に設定されているので、*plot2* のウィンドウは *plot1* のウィンドウより大きくなっています。このリソースにアクセスするために、次の一行を加えます。

```
#include <X11/StringsDefs.h>
```

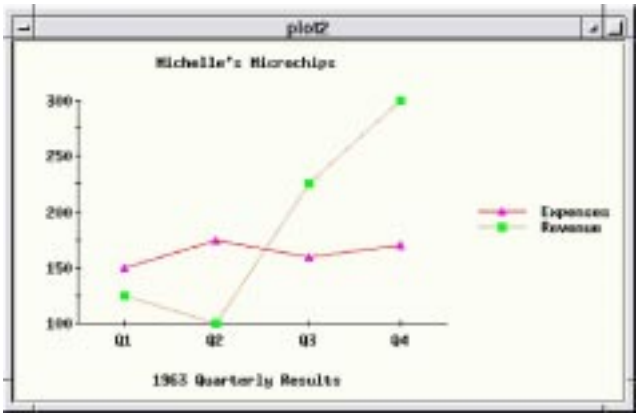


図5　plot2 のウィンドウ

1.7　棒グラフに変更する

XRT/graph の優れた機能の一つは、リソースを個別に自由に変更できることです。[1] 棒グラフから線、円グラフへといつでもグラフ・タイプを変更することができます。

bar2.c を作成するために、*plot2.c* の *XtVaCreateManagedWidget()* コールに以下を付け加えます。

```
XtNxrtType、XRT_TYPE_BAR
```

図6は、*bar2.c* がコンパイル、実行された結果です。

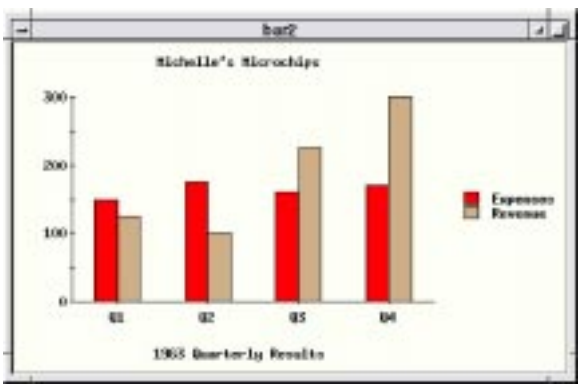


図6　Bar2 のウィンドウ

[1] 幾つかのリソースでは相互依存がありますが、リソースの大部分は完全に独立しています。相互依存性については、Appendix A をご覧ください。

1.8 グラフのインバート

ほとんどのグラフは、X 軸が横、Y 軸が縦になっていますが、これを入れ替えた方が適当な場合もあります。

線グラフをインバートすると、Y 値は横軸に、X 値は縦軸に対して表示されます。棒グラフでは各棒が横向きになります。

XRT/graph のプログラミングでは、どちらの軸が横または縦に表示されるかは `XtNxrtInvertOrientation` の値で決まります。

XtNxrtInvertOrientation

軸を入れ替えるには、`XtNxrtInvertOrientation` のリソースを `True` に設定してください。デフォルトは `False` です。

図 7 は、*plot2.c* と *bar2.c* の `XtNxrtInvertOrientation` を `True` に設定した結果です。

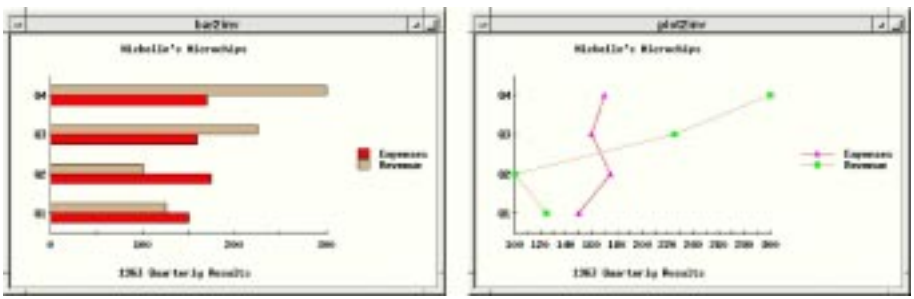


図 7 `XtNxrtInvertOrientation` が `True` の *plot2* と *bar2*

1.9 グラフを完成させる

panel.c では、この章の内容を合わせて一つのプログラムにします。(*panel.c* は、*$XRTHOME/src/graph/example* にあります。)図 8 は、*panel.c* をコンパイル、実行した結果です。

ユーザは、ウィンドウの上部にあるトグル・ボタンをクリックすることでデータの表示法を変更することができます。

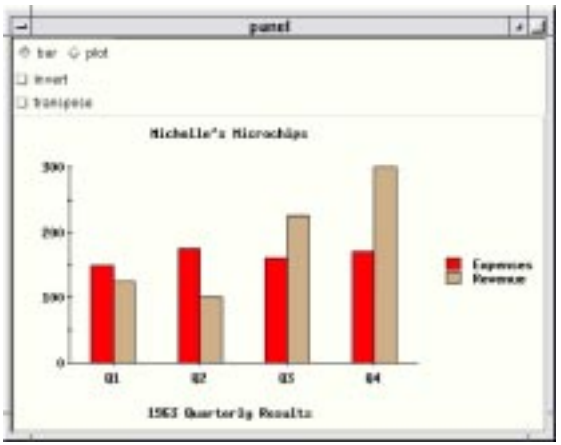


図8 *panel* のウィンドウ

各トグル・ウィジェットは、グラフのリソースを変更するコールバック手続きを持っています。

例えば、インバート・トグルのコールバック手続きは、`XtNxrtInvertOrientation` リソースを以下のように設定します。

```
void
invert_proc(
  Widget   w,
  XtPointer    client_data,
  XtPointer    call_data
  )
{
  XmToggleButtonCallbackStruct *cbs;
  cbs = (XmToggleButtonCallbackStruct *) call_data;
  XtVaSetValues(graph,
    XtNxrtInvertOrientation, cbs->set,
    NULL);
}
```

1.10 サンプル・コード

XRT/graphには、幾つかの例やデモのプログラムが提供されています。例は、特定な機能を使用した短いプログラムです。デモは、複数の機能を使用した、アプリケーションです。

例

これらは、*$XRTHOME/src/graph/examples*にあります。

- *bar2tran.c* と *bar2tran_cpp.cxxbubble.c*
- *common.c*

panel.c

pie.c

- *plot1.c*
- *radar.c*
- *scrollgraph.c*
- *simple_uil.c*
- *slider.c*
- *stock.c*
- *time.c*

デモ

デモのアプリケーションは、*$XRTHOME/src/graph/demos*にあります。各デモごとにディレクトリがあります。

- *builder* は、XRT/graph のプロパティ・エディタ(Property Editor)を使用しています。グラフの表示をインタラクティブに変更したり、(グラフを記述した).XRT ファイルを見ることができます。
- *drill_down* は、多くのデータを管理し、異なったレベルの精度で表示します。複数のグラフを用いて、前にビューしたグラフ、メインのグラフ、凡例を表示します。
- *feedback* は、ユーザ・イベントのマップやピックとそれらのイベントの視角的なフィードバック表示のテクニックを解説します。またマーカの使用法も示しています。
- *flow* はパルス・チャートの一種です。色でデータ値の範囲を区別する例を示しています。データのグラフ化と凡例の表示の二つのグラフ・オブジェクトを使用しています。凡例はポップアップ・ウィンドウに表示されています。
- *hilow* は二つの株価の high-low チャートです。ユーザの要求に対してデータ・オブジェクトを操作する方法を例示しています。
- *polar* は、polar リソースを使用して極データを表示します。データの生成に幾つかの数式を使用しています。
- *scroll_time* は、時間軸の機能の使用法のデモで、表示するデータのサブセットの選択に、*Xlib* ルーチンを使用しています。

- *stripper* は、リアルタイムのストリップ・チャートを作成するためにどのように高速アップデートを使用するか示しています。入ってくるデータの速さとスクロール・バックの割合を調節できます。
- *text_area* は、線グラフ、棒グラフ、積上げ棒グラフ、レーダー・チャート、円グラフにどのようにテキスト領域を作成し貼り付けるか示します。このデモの開始ページの背景にはイメージが表示されています。
- *vmgraph* は、使用されているシステムの仮想メモリ統計を秒単位でアップデートされる三つのグラフで表示します。パイプからのデータの読み込み法も示しています。このデモには *vmstat* プログラムが必要ですが、システムによっては使用できない場合もあります。

複数の XRT 製品を使用したデモ

- *graph_editor* は、グラフ・データの表示に XRT/table を使用しています。
- *mvc* は、XRT/graph、XRT/table、XRT/3d を使用した、XRT MVC Kit のデモです。
- *profile* は、XRT/3d によって表示される 3D サーフェスの 2D 断面図を XRT/graph で表示します。

2

XRT/graph の基本操作

Motif との統合 ■ 用語の解説
グラフ・タイプ ■ 浮動小数点のリソース
UseDefault リソース ■ ポインタ・リソース
ストリング・リソース ■ 色
リソースファイルの使用

2.1 Motif との統合

XRT/graph は、Motif ツールキットに新たなウィジェットを追加します。このウィジェットは、他の標準のウィジェットと同様に Xt Intrinsics で操作することができます。

Xt Intrinsics は、ウィジェットをクラス編成するオブジェクト指向アーキテクチャを定義します。例えば、Motif XmPushButton ウィジェットは、Xt Core ウィジェットのサブクラスの Xm Primitive のサブクラスの XmLabel のサブクラスであり、XRT/graph ウィジェットは XmPrimitive のサブクラスとなります。これは、グラフに、XRT/graph で定義されるリソースの他に XmPrimitive と Core のリソースも設定できることを意味します。例えば、(XmPrimitive で定義される)`XmNshadowThickness` リソースは、グラフを囲む線を調整することができます。また、(Core で定義される)`XtNwidth` と `XtNheight` リソースは、グラフの幅と高さを調整することができます。

2.2 用語の解説

図 10 は、XRT/graph の構成です。グラフには主に四つの領域(ヘッダー、フッター、凡例、グラフ)があり、それぞれ起点、幅、高さを持ちます。図 10 では、各領域の枠線が、装飾無しの 2 ピクセル幅の線で描かれています。

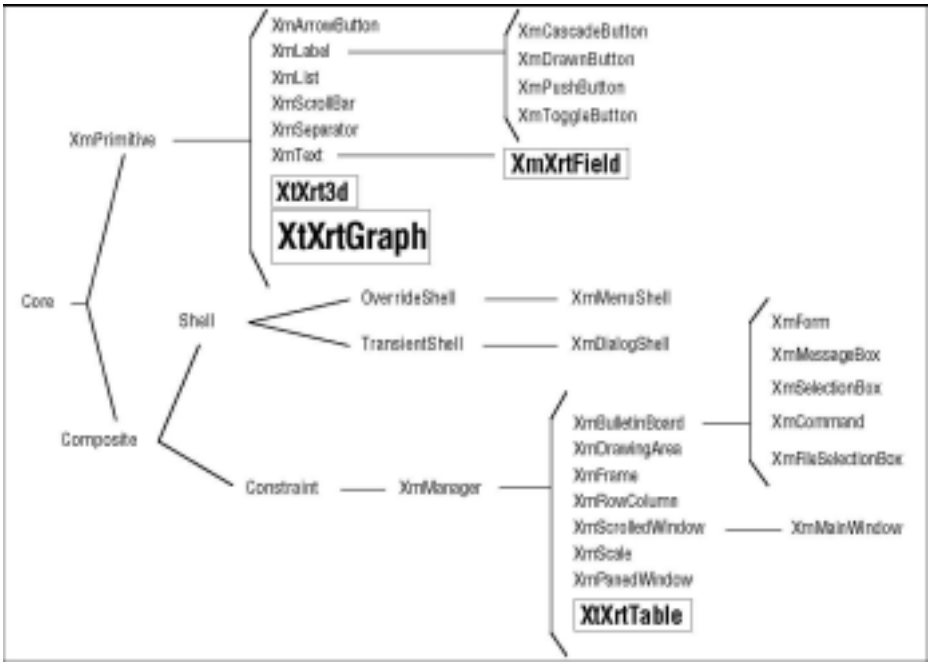


図9　Motif クラス階層内のグラフ・ウィジェットの位置

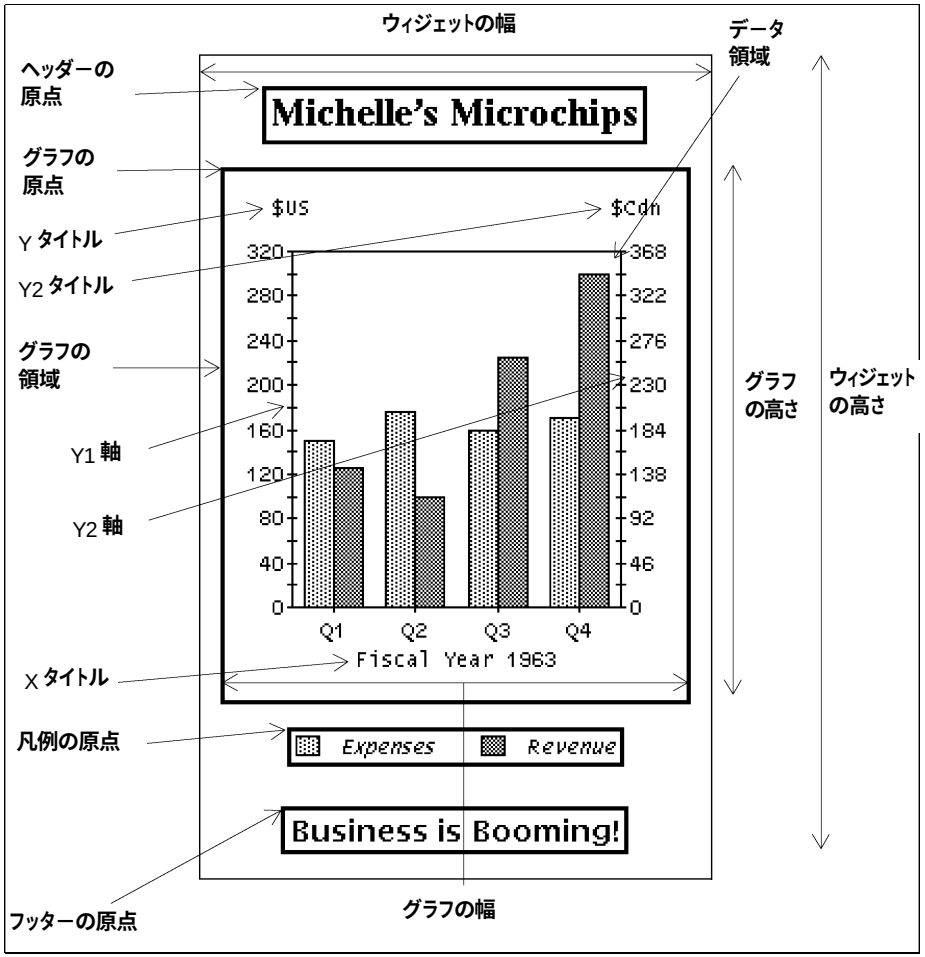
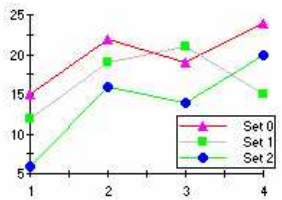
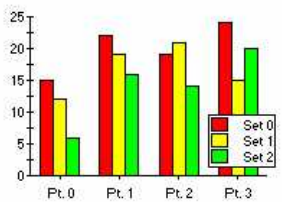
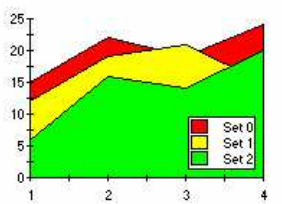
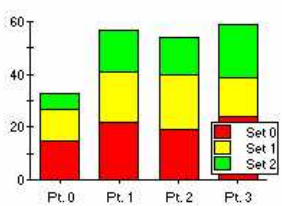

ウィジェットの幅
データ
領域
ヘッダーの
原点
Michelle's Microchips
グラフの
原点
$US
$Cdn
Y タイトル
Y2 タイトル
320
280
240
200
160
120
80
40
0
368
322
276
230
184
138
92
46
0
グラフの
領域
グラフ
の高さ
ウィジェット
の高さ
Y1 軸
Y2 軸
Q1
Q2
Q3
Q4
Fiscal Year 1963
X タイトル
凡例の原点
Expenses
Revenue
Business is Booming!
フッターの原点
グラフの幅

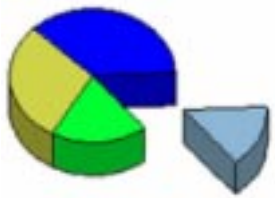
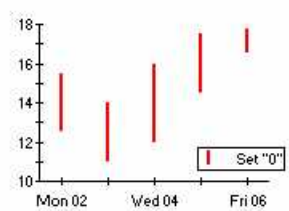
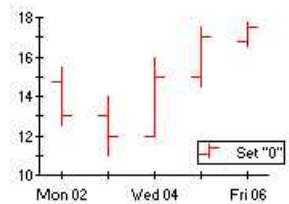
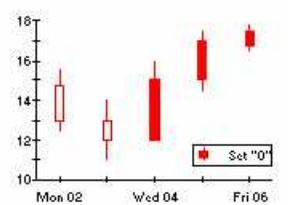
図 10　グラフ領域

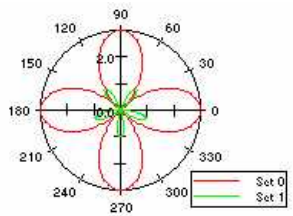
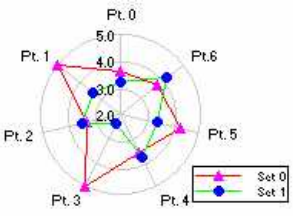
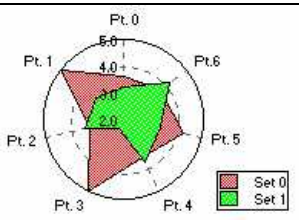
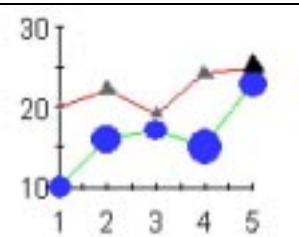
2.3 グラフ・タイプ

XRT/graph には、基本で 12 種類のグラフが用意されています。この基本タイプを応用して更に複雑なグラフが作成できます。

グラフ・タイプは、`XtNxrtType` リソースで指定します。このリソースは、データや他のリソースからは独立して変更できますので、同じデータを異なるグラフで表示することができます。次の表には、基本的なグラフ・タイプ、グラフ化できるデータ型と各グラフの特徴がまとめてあります。

	グラフ例	配列 (Array) データ	一般 (General) データ	特徴
1	 XRT_TYPE_PLOT	✓	✓	**線** データ・ポイントを線で結ぶ。 配列データの時は、X 値がセットによって共有される `XrtDataStyle` の線パターン、幅 色 ポイント・スタイル、サイズで表示をコントロールする 配列データの時は、積み上げと 100%軸のグラフが作成可能
2	 XRT_TYPE_BAR	✓		**棒** 各セットを棒で表す。クラスタ(cluster、棒のかたまり)の数はデータ・ポイントの数と一致する。各クラスタは、各セットの n 番目のデータを表示する。 X 軸は、通常 ポイント・ラベルで選択される `XrtDataStyle` の塗りつぶしパターンと色で表示をコントロールする
3	 XRT_TYPE_AREA	✓	✓	**面** データ・ポイントの線を結び、囲まれる部分を塗りつぶす。各セットは、重なり合う。 配列データの時は、X 値がセットによって共有される `XrtDataStyle` の塗りつぶしパターンと色で表示をコントロールする 配列データの時は、積み上げと 100%軸のグラフが作成可能
4	 XRT_TYPE_STACKING_BAR	✓		**積み上げ棒** 各セットを積み上げられた棒の一部として表す。クラスタの数は、データ・ポイントの数と一致する。各クラスタは、各セットの n 番目のデータを表示する。負の Y 値は、X 軸の下に表示される。 X 軸は、通常 ポイント・ラベルで選択される `XrtDataStyle` の塗りつぶしパターンと色で表示をコントロールする 100%軸のグラフが作成可能

	グラフ例	配列 (Array) データ	一般 (General) データ	特徴
5	 XRT_TYPE_PIE	✓		**円** 各セットを円のスライスとして表す。円の数は、データ・ポイントの数と一致する。("しきい値"より小さい値は、"その他"のスライスにグループ化される。) スライスは、円から切り離して表示可能。各円は、各セットのn番目のデータを表示する。 円は、ポイント・ラベルでのみ選択される XrtDataStyle の塗りつぶしパターンと色で表示をコントロールする
6	 XRT_TYPE_HILO	✓		**Hilo**(P84、4.12 参照) 二つのセットを"high-low"の棒として表す。各セットのポイントは、棒のコンポーネントを定義する。 一番目のセット - ポイントは"high"値 二番目のセット - ポイントは"low"値 セットの各ペアは、一つの論理的なセットを構成し、セットに指定する手続きやリソースが使用できる[a] ペアの内 最初のセットの XrtDataStyle の線パターン、幅 色で表示をコントロールする
7	 XRT_TYPE_HILO_OPEN_CLOSE	✓		**HiloOpenClose**(P84、4.12 参照) Hilo と類似しているが四つのセットを一緒にして、"high-low-open-close"の棒として表す。一、二番目のセットは、Hilo と同じ役割を持つ。 三番目のセット - ポイントは"open"値 四番目のセット - ポイントは"close"値 四つのセットを一つにまとめて論理的なセットを構成し、セットに指定する手続きやリソースが使用できる[a] 四つのセットの内 最初のセットの XrtDataStyle の線パターン、幅 色 ポイント・サイズで表示をコントロールする
8	 XRT_TYPE_CANDLE	✓		**キャンドル**(P84、4.12 参照) HiLoOpenClose の特別な形。四つのセットで、キャンドルを構成する。 四つのセットを一つにまとめて論理的なセットを構成し、セットに指定する手続きやリソースが使用できる[a] 最初のセットの XrtDataStyle の線パターン、塗りつぶしパターン、幅 色 ポイント・サイズで表示をコントロールする 各セットの XrtDataStyle によっても表示をコントロールできる

	グラフ例	配列 (Array) データ	一般 (General) データ	特徴
9	 XRT_TYPE_POLAR	✓	✓	**ポーラー**(P87、4.13 参照) 極座標(theta、r)にデータ・ポイントを結んで表す。X 値は回転、Y 値は原点からの距離を表す。 配列データの時は、X 値はセットによって共有される X 軸の範囲はセットできず、Y 値の範囲はデータ範囲内にはセットできない XrtDataStyle の線パターン、幅 色 ポイント・スタイルとサイズによって表示がコントロールされる 配列データの場合は、積上と 100%軸グラフの作成が可能
10	 XRT_TYPE_RADAR	✓		**レーダー**(P87、4.13 参照) 各セットをレーダーの軸に沿って、データ・ポイントを結んで表す。各セットの n 番目のポイントの Y 値が各軸上に表示される。 X 軸は、ポイント・ラベルまたは値ラベルで表示される XrtDataStyle の線パターン、幅 色 ポイント・スタイルとサイズで表示をコントロールする 積上と 100%軸グラフの作成が可能
11	 XRT_TYPE_FILLED_RADAR	✓		**Filled(塗りつぶし)レーダー**(P87、4.13 参照) データ・ポイントを結び内側を塗りつぶす。各セットは重なり合う。 X 軸は、ポイント・ラベルまたは値ラベルで表示される XrtDataStyle の塗りつぶしパターンと色で表示をコントロールする 積上と 100%軸グラフの作成が可能
12	 XRT_TYPE_BUBBLE	✓		**バブル**(P84、4.12 参照) 線グラフの要素に、さらに各ポイントのシンボルの大きさや場所を示す情報が加わったグラフ。 論理的なセットは、二つの実際のセットで構成される二つの内 最初のセットの XrtDataStyle の線パターン、幅 色 ポイント・サイズで表示をコントロールする

a: データ値とデータ・スタイルは例外です。これらは、値または棒の表示のカスタマイズを可能にするために、論理的なセットではなく、実際のセット・インデックスを使用します。

2.4 浮動小数点のリソース

XtSetValues() または *XtVaSetValues()* で浮動小数点のリソースを設定するときは、*XrtFloatToArgVal()*を使用してください。[1] 例えば、現在の Y 軸の最大値を 2 倍にするには、次のようにしてください。

```
float ymax;
XtVaGetValues(mygraph,
  XtNxrtYAxisMax, &ymax, NULL);
ymax = ymax * 2.0;
XtVaSetValues(mygraph,
  XtNxrtYAxisMax, XrtFloatToArgVal(ymax),
  NULL);
```

2.5 UseDefault リソース

多くの XRT/graph リソースは、対応する `UseDefault` リソース(例:`XtNxrtYNum` と `XtNxrtYNumUseDefault` 等)を持っています。

`UseDefault` リソースは、XRT/graph がリソースのデフォルト値を計算するかを決定する Boolean 関数です。

例えば、`XtNxrtYNumUseDefault` が `True` であれば、XRT/graph がグラフ作成時にグラフ領域のサイズ、`XtNxrtYPrecision` の値等を考慮して、最も適切なデフォルト値を決定します。もし、`XtNxrtYNumUseDefault` が `False` であれば、XRT/graph は一番最後に指定された `XtNxrtYNum` 値を使用します。

対応するリソースが設定されていない場合や、XRT/graph によって計算されていない場合は、`UseDefault` の `False` 設定は無視されます。これらのリソースを設定すると、副作用で `UseDefault` リソースが `False` にセットされます。

以下のコードは、`XtNxrtYNum` の値を現在の値に強制的に変更します。この副作用で `XtNxrtYNumUseDefault` は `False` になります。

```
float ynum;
XtVaGetValues(graph, XtNxrtYNum, &ynum, NULL);
XtVaSetValues(graph,
  XtNxrtYNum, XrtFloatToArgVal(ynum),
  NULL);
```

次のコードでデフォルトに戻り、XRT/graph はグラフ作成時に `XtNxrtYNum` のデフォルト値を計算します。

```
XtVaSetValues(graph,
  XtNxrtYNumUseDefault, True,
  NULL);
```

リソースファイルの `UseDefault` の設定は無視されます。

---

[1] *XrtSetValues()*を使用して、浮動小数点リソースをセットしている場合は、この関数は必要ありません。

2.6 ポインタ･リソース

XRT/graph のリソースの多くは、*XtGetValues()*で使用されると、構造体か文字のポインタを返します。このポインタは、メモリ内のデータを変更するためには使用できません。ポインタが指すものは全て"read only"です。データのコピー作成には、*XrtDupDataStyles()*や*XrtDupStrings()*の簡易ルーチンが使用できます。これらのコピーは、*XrtFreeDataStyles()*や*XrtFreeStrings()*で解放されます。

但し、`XtNxrtData` と `XtNxrtData2` は例外です。これらのリソースは、常にアプリケーションのメモリに存在するデータ･オブジェクトを指しています。

XRT/graph は、ポインタ･リソースを通して渡された全てのデータのコピーを作成します(`XtNxrtData` と `XtNxrtData2` は例外です)。

例えば、最初のヘッダー･ストリングとフッター･ストリングを交換する場合、次のコードが使用できます。

```
char **hs, **fs, **myhs, **myfs, *tmp;
XtVaGetValues(graph, /* Get read-only ptrs */
  XtNxrtHeaderStrings, &hs,
  XtNxrtFooterStrings, &fs,
  NULL);

myhs = XrtDupStrings(hs); /* Make copies */
myfs = XrtDupStrings(fs);

tmp = myfs[0]; /* Interchange 1st */
myfs[0] = myhs[0];
myhs[0] = tmp;

XtVaSetValues(graph, /* Reset them */
  XtNxrtHeaderStrings, myhs,
  XtNxrtFooterStrings, myfs,
  NULL);

XrtFreeStrings(myhs); /* Free my copies */
XrtFreeStrings(myfs);
```

2.7 ストリング･リソース

リソースの幾つか(`XtNxrtXTitle`、`XtNxrtPointLabels` 等)は、ストリングかストリングの配列を値として取ります。

これらのリソースには表示できる文字だけを指定してください。ASCII 制御文字('\n'等)を含むストリングを表示しようとすると、異常な結果が得られることがあります。

もし、アプリケーションがデータの取得に *fgets()*のような標準のライブラリのルーチンを使用している場合、そのストリングを XRT/graph のリソース値とする前に、各ストリングから \n を取り除いてください。

ストリングの配列を値として取るリソースをセットする時は、最後のストリング･ポインタが `NULL` であることを確認してください。XRT/graph は、リストのターミネートにこの `NULL` ポインタを必要とします。尚、`XtGetValues` によって取得されたストリング配列は自動的に `NULL` で終了します。

2.8 色

XRT/graph では、ウィンドウの背景、前景やグラフの線、塗りつぶし、ポイントの色を指定することができます。

XRT/graph はデフォルトの色の選択もしますので、簡単なアプリケーションでは色の指定の必要はありません。

色の指定

色は以下のように、色名か 16 進数の RGB{赤(R)、緑(G)、青(B)}構成を含むストリングで指定します。

```
char * my_named_color = "LemonChiffon";
char * my_rgb_color = "#228B22";
```

色名は、X Window サーバのカラー・データベースから選択してください。通常、UNIX システムでは、*/usr/lib/X11/rgb.txt* ファイルにあります。

RGB 値で指定する場合、ストリングは'#'で始め、次に RGB(赤、緑、青)の二文字の 16 進数値が来ます。上の例の `my_rgb_color` は赤が 0x22、緑が 0x8B、青が 0x22 です。[1]

XRT/graph がモノクロで表示される場合、グラフの領域や前景の白以外の色は黒に、背景の黒以外の色は白に表示されます。

一度グラフが作成されると、アプリケーションのカラーマップはアップデートや変更はできません。グラフの色の変更は、グラフのリソースのアップデートで行います。(例: `XtNxrtBackgroundColor`、`XtNxrtDataStyles` 等)

カラーマップのフラッシュ

X Window サーバのカラーマップは共用リソースです。ある状況においては、全てのアプリケーションで表示中の色の数が、使用可能なカラーマップのエントリー数より多いことがあります。この場合、幾つかのウィンドウでユーザがマウスを動かした時、その色が他の色に変更(フラッシュ)してしまいます。

サーバがスタティック(静的)な色だけを提供する場合は、このカラー・フラッシュが起こりません。アプリケーションには、指定通りの色ではありませんが、ウィンドウのカラーマップの中で最も近い色が選択されます。

[1] 色の指定の構文は例の通りでなくてもよく、Xlib 手続きの *XParseColor()*がサポート可能なストリングならどれでも使用できます。

しかし、ほとんどのアプリケーションではダイナミック(動的)な色を必要とします。この場合は、アプリケーションに指定通りの色を提供することができますが[1]、サーバの使用可能な色の数や実行中のアプリケーションの数と状態によって、カラーマップ・フラッシュが起こります。

ダイナミック・カラーマップで実行中、XRT/graph は、アプリケーションの要求に従って自動的にサーバのカラー・セルの割り当て、または割り当て解除を行い、カラーマップ・リソースへの要求を最小限にします。もし、XRT/graph アプリケーションが使用可能な数以上の色を要求すると、使用不可な色の部分は、白黒で表示されます。

一番近い色を取得する

XRT/graph は、色のリソースにストリングではなく pixel を使用し、*XAllocColor()* と *XFreeColors()* によって色の管理を行います。カラーマップが足りなくなると、XRT/graph は、白または黒を割り当てますが、`XtNxrtColorFunc` リソースを使用することによって、独自の色の割り当て方法を指定することができるようになります。

色の指定には、この新しいリソースの他、(XPM のカラー・マッチングのアルゴリズムを適用した)ファンクションがサンプルとして提供されています。このソースコードは、*$XRTHOME/src/graph/util/x_color_func.c* にあり、*XrtSampleColorFunction()* として XRT/graph のライブラリにも含まれています。その他、独自の色のファンクションを使用することもできます。

`XtNxrtColorFunc` はデフォルトでは `NULL` です。この場合、前のバージョンと同様に balck/white のアルゴリズムを使用します。`NULL` でない場合は、このファンクションを呼び出し、そこで指定する色が使用されます。

独自のファンクションを使用する場合は、必ず下記の条件に従ってください。

1. 使用するファンクションは、次ページのプロトタイプを持つ
2. XRT/graph は、ウィジェットが破棄されたり、ユーザが *XtVaSetValues()* によって色を変更した時には、`XFreeColors` によって色を解放するので、色は(*XAllocColor()* を使用して)必ず割り当てる。色を割り当てることができない場合は、`NO_PIXEL_VALUE` を返すようにする。この場合、ファンクションが指定されていなければ、XRT/graph は黒または白を割り当てる。
3. ファンクションがメモリを割り当てたり、解放したりする必要がある時は、サンプルのファンクションと同じ方法で(do-free flag を使用して)それらを行う。
4. 色のパラメータが `NULL` であれば、色を割り当てずに、`NO_PIXEL_VALUE` を返すようにする。(XRT/graph は、ウィジェットが破棄された時に、color = `NULL`、do-free=`True` でこのファンクションを呼び出す。)

[1] ハードウェアの表示能力の範囲内。

色のファンクションのプロトタイプ

```
unsigned long (*xrt_color_func)(
      Display      *display,
      Int      screen,
      Colormap     colormap,
      Xcolor       *color,
      Boolean      do-free
      )
```

2.9 リソースファイルの使用

リソースの値を設定するには、C コードより、リソースファイルを使用した方がよい場合があります。リソースファイルには次の利点があります。

スピード - コンパイルしないので、変更が瞬時に確認できます。

柔軟性 - エンドユーザがニーズにあわせてリソースの値を変更できます。

`UseDefault` リソースの例外を除いて、XRT/graph の全てのリソースはリソースファイルで設定ができます。但し、ほとんどのリソースはエンドユーザが変更を加えたりするには複雑すぎるかもしれません。[1] アプリケーションの全てのウィジェットのリソースファイルは、組み合わせてアプリケーションのデフォルト・ファイルを作成することができます。

リソースファイルとその使用法の詳しい内容は、Xt Intrinsics のマニュアル等をご覧ください。Appendix I には、リソース設定の構文の解説と、サンプルのリソースファイルがあります。

[1] XRT/graph のプロパティ・エディタ(Property Editor)を使用する場合は、[Application Defaults]のダイアログで、各リソースがどのように定義されるか(C コードかまたはリソースファイルを使用するか)を選択することができます。

3

プロパティ・エディタの使用

はじめに ■ プロパティ・エディタの使用

プロパティ・エディタのプログラミング ■ ソースコードの生成

XRT/graph のプロパティ・エディタ(Property Editor)は、プロトタイプ、開発、アプリケーションの維持に使用できるライブラリです。このライブラリは、アプリケーションのエンドユーザが使用できる、グラフのプロパティ・ エディタとしても提供することができます。プロパティ・エディタは、XRT/graph に次の機能を追加します。

- アプリケーションの実行時に、グラフのリソースのインタラクティブな編集を可能にする Compact Property Editor。
- アプリケーションの一部としてエンドユーザにカスタマイズされたプロパティ・エディタを提供できる機能。ユーザがいつそれを使用するか、ユーザが.XRT ファイルまたはデータ・ファイルを読み込みまたは保管することができるかどうかをコントロールすることができます。
- (.XRT ファイルと呼ばれる)グラフの記述をテキスト・ファイルに保存する。
- .XRT ファイルから C のソースとリソース・ファイルを生成する。C コードとリソース・ファイルは、リソースを.XRT ファイルで設定されている値にセットして、グラフ・ウィジェットをイニシャライズします。

プロパティ・エディタを使用して、開発にかかる時間を大幅に削減することができます。グラフは、コードを書かずにプロトタイプを作成したり、アプリケーション実行中にリソース設定のテストを行い、その場でその結果を確認することができます。

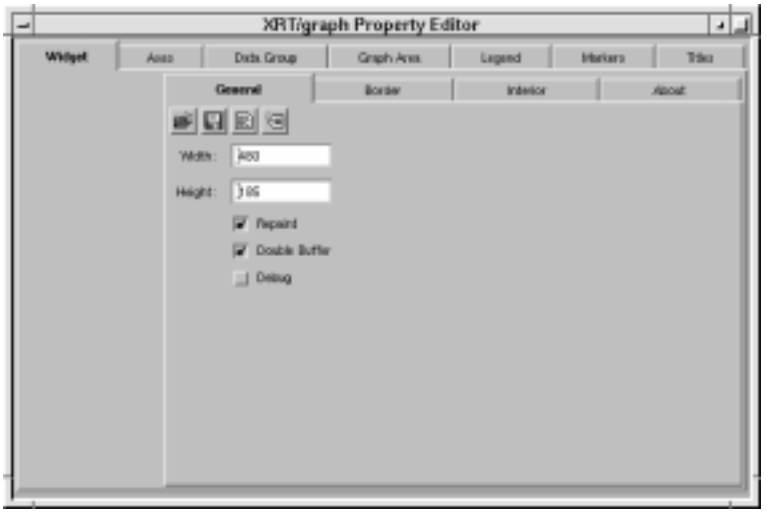


図11　XRT/graph のプロパティ・エディタ

3.1　はじめに

XRT/graph のプロパティ・エディタは、*builder* というデモからアクセスしたり、このデモをアプリケーションのプログラムに追加することによりアクセスすることができます。どちらの方法も特有の利点があります。

- *builder* デモによって、グラフを視覚的に構築したり、サンプルの.XRT(グラフ記述)ファイルのグラフを見ることができます。
- プログラムから *builder* にアクセスすることにより、リソース設定のデバッグを行ったり、プログラムの実行中に異なるリソース設定をテストしたりすることができます。

Builder を使用する

プリコンパイルされた *builder* デモを実行するには、"`builder`"と入力してください。XRT/graph ウィジェットのウィンドウが表示され、プロパティ・エディタがポップアップします。このデモのソースコードは、*$XRTHOME/src/graph/demos/builder* にあります。

アプリケーションで使用する

プロパティ・エディタをアプリケーションで使用する場合は、これをアプリケーションに明示的に追加する必要があります。プロパティ・エディタは、別のライブラリとして提供されていますので、開発が終了した時に、簡単に削除することもできます。このライブラリをアプリケーションに追加するには、次の要領に従ってください。

アプリケーションのリンク時には、プロパティ・エディタ(*libxrtmprop.a*)や他の XRT 製品(XRT/table、XRT/gear、XRT/field)や XPM ライブラリの前に、XRT/graph のライブラリを置いてください。(*builder* デモの Makefile を参照してください。)
プロパティ・エディタを使用する場合は、これらの製品のライセンスは必要ありませんので、*xrt_auth* のライセンスに関するメッセージは無視してください。

グラフ作成後に *XrtPropertyEditorEnable()*を呼び出します。

```
graph = XtVaCreateManagedWidget("graph",
  xtXrtGraphWidgetClass, toplevel,
  NULL);
...
XrtPropertyEditorEnable(graph, True, NULL, True, True);
```

*builder*デモからfallbackリソースをアプリケーション・プログラムにコピーしてください。これらのリソースは、プロパティ・エディタのダイアログのサイズや色をコントロールします。

アプリケーションを保管、コンパイルし、実行してみてください。

グラフ・ウィジェット上で、マウスの右ボタンをクリックすると、プロパティ・エディタが表示されます。

3.2 プロパティ・エディタの使用

グラフ上で、マウスの右ボタンをクリックし、プロパティ・エディタを表示します。但し、アプリケーションでは、プロパティ・エディタが使用できるようにセットされ、必要なライブラリがインクルードされている必要があります。(3.1 参照)

グラフ・プロパティの編集

編集したいプロパティに該当するタブを選択します。タブの中には、更に複数のタブを持つものがありますので、それらを選択しながら編集したいプロパティをみつけます。

左側にツリー・コントロールが表示されていたら、そのリストより、編集するオブジェクトを選択します。

プロパティの値を編集してください。

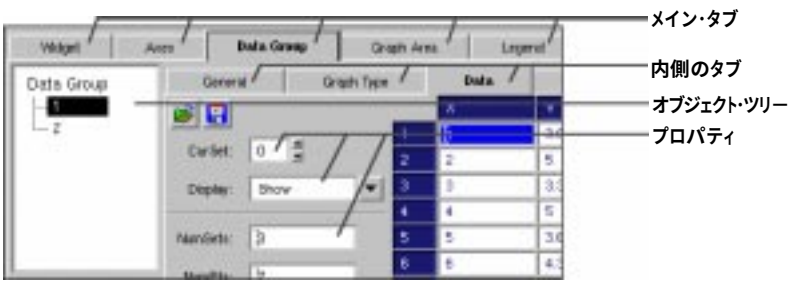


図12　プロパティ・エディタのエレメント

プロパティ・エディタでリソース値を変更すると、その変更はグラフに直ぐに適用され、表示されます。[1] 変更が瞬時に確認できるので、プロパティ・エディタを離れることなく、

[1] [Widget]タブの[General]にある、[Repaint]のプロパティがチェックされている場合

そのまま他の変更も加えることができます。プロパティが変更された後は、それを取り消す(undo)には、もう一度そのプロパティを元の値に設定し直してください。

必要なプロパティの変更が終了したら、Window マネージャのコントロール・メニューから[Close]を選択してください。

.XRT ファイルのロードと保管

[Widget]タブの[General]にあるツールバー・ボタンを使用して、.XRT ファイルをロードまたは保管します。サンプルのグラフは、*$XRTHOME/src/graph/builder* にあります。

グラフのデータ・ファイルのロードと保管

[Data Group]の[Data]にあるツールバー・ボタンを使用して、データ・ファイルをロード、保管します。データ・ファイルは、必ず、*XrtDataCreateFromFile()*が理解できるフォーマットにしてください。(Appendix D 参照) サンプルのデータ・ファイルは、*$XRTHOME/src/graph/data* にあります。

制限事項

プロパティ・エディタは、XRT/graph の全てのリソースにアクセスできますが、テキスト・オブジェクトのリソースは例外です。また、`XtNxrtData` リソースが `XrtData` 構造体を指している場合は、データの編集はできません。

3.3 プロパティ・エディタのプログラミング

プロパティ・エディタのライブラリには、幾つかの手続きが提供されています。これらの手続きを使用するアプリケーションは、必ず、プロパティ・エディタのライブラリとリンクしている必要があります。(3.1 参照)

3.3.1 エンドユーザのプロパティ・エディタのカスタマイズ

アプリケーションのエンドユーザにプロパティ・エディタを提供する場合、*XrtPropertyEditorEnable()*を使用して、エディタの機能をカスタマイズしたり不可に設定することができます。*XrtPropertyEditorEnable()*は次のプロトタイプを持っています。

```
Boolean
XrtPropertyEditorEnable(
  Widget  graph,
  Boolean enable,
  char*title_string,
  Boolean allow_load_save,
  Boolean do_updates
  )
```

不可に設定するには、*enable* を `False` と渡してください。プロパティ・エディタのタイトル・バーに表示されるテキストを変更するには、表示するタイトルを *title_string* 引数として指定してください。ユーザが.XRT ファイルやデータ・ファイルをロード/保管できないようにするには、*allow_load_save* を `False` にしてください。(この操作により、[Widget]の[About]タブにある KL Group 社のロゴも削除されます。)

3.3.2 ポップアップとポップダウン

XrtGraphProp.h ヘッダー・ファイルに定義されている関数を使用して、プロパティ・エディタの表示、非表示、アップデートのコントロールを行うことができます。これらの関数を使用するには、アプリケーションは、"`#include XrtGraphProp.h`"とし、XRT/graph と Property Editor のライブラリ、Xrt/field、XRT/gear、XRT/table とリンクさせてください。

XrtPopupPropertyEditor()

プロパティ・エディタを表示します。

```
Boolean
XrtPopupPropertyEditor(
  Widget  graph,
  char*title_string,
  Boolean allow_load_save
  )
```

title_string は、タイトル・バーに表示されるテキスト・ストリング。*allow_load_save* は、ユーザが、.XRT またはデータ・ファイルをロード/保管し、C コードを生成できるようにするか指定します。

XrtPopdownPropertyEditor()

プロパティ・エディタを閉じます。*destroy* が `True` でない限り、エディタは破棄されません。

```
Boolean
XrtPopdownPropertyEditor(
  Widget  graph,
  Boolean destroy
  )
```

XrtUpdatePropertyEditor()

現在のグラフのリソース値で、プロパティ・エディタをアップデートします。

```
Boolean
XrtUpdatePropertyEditor(
  Widget  graph
  )
```

3.3.3 グラフの記述のロードと保管

*XrtLoadGraphFromFile()*を使用して、グラフの記述をロードし、現在のリソース値を置き換えることができます。現在のリソース値は、*XrtSaveGraphToFile()*を使用して.XRT ファイルに保管することができます。これらの手続きは、*XrtGraphInv.h* で定義され、次のプロトタイプを持っています。

```
Boolean
XrtLoadGraphFromFile(
  Widget  graph,
  char*filename,
  char*msg
  )
```

```
Boolean
XrtSaveGraphToFile(
  Widget  graph,
  char*filename,
  char*msg
  )
```

filename はXRT ファイルのパス名です。エラーが起きると `False` を返し、エラー・メッセージを *msg* に書きます。*msg* が `NULL` であれば、エラーは *stderr* に書かれます。*msg* は最低100 byteです。

これらの関数を使用するには、アプリケーションで"`#include XrtGraphInv.h`"とし、プロパティ・エディタのライブラリ(*libxrtmprop.a*)にリンクしてください。

3.3.4 プロパティ・エディタのソースコード

プロパティ・エディタは、ソースコードとして提供されていますので、アプリケーションの要求に応じてカスタマイズすることができます。但し、カスタマイズされたプロパティ・エディタをコンパイルするには、XRT PDS のライセンスか、または XRT/table、XRT/gear、XRT/field のライセンスが必要です。

プロパティ・エディタのソースコードは、*$XRTHOME/src/graph/properties* にあります。

3.4 ソースコードの生成

.XRT ファイルを作成した後に、アプリケーションに使用できる C コードを生成することができます。ソースコードは、.XRT ファイルで定義されたリソース値にセットされたグラフ・ウィジェットを作成します。各リソースは、プロパティ・エディタの[Applicaiton Defaults]で、C またはリソース・ファイルのどちらでセットするか指定できます。

ソースコードの生成には、次の三つの方法があります。

- *xrt_gmf* を実行する
- プロパティ・エディタを使用する
- *XrtGenerateCode()*を使用して直接アプリケーションから作成する

xrt_gmf プログラム

xrt_gmf を実行するには、"`xrt_gmf <filename>`"と入力してください。<`filename`>の場所には、.XRT ファイルが来ます。(".XRT"の部分は入力する必要はありません。*xrt_gmf* は、実行時に次のようなメッセージを出力します。

```
% xrt_gmf mygraph
Reading mygraph.XRT
Writing mygraph_xrt.c and mygraph_xrt.res
Writing mygraph_xrt.h
Writing mygraph_xrt_main.c
Writing mygraph_xrt.make
```

ANSI C または C++アプリケーションで使用するコードを生成する場合は、*xrt_gmf* を実行する時に*-a* と指定してください。

プロパティ・エディタ

[Widget]の[General]タブにあるボタン(下記の図参照)をクリックして、アプリケーションのデフォルト・ファイルとコードを生成します。



プロパティ・エディタは、ANSI C コードのみを生成します。

XrtGenerateCode()

*XrtGenerateCode()*は、指定された.XRT ファイルからソースコードを生成します。この手続きは、*XrtGraphInv.h* に定義され、次のプロトタイプを持ちます。

```
Boolean
XrtGenerateCode(
  Widget  graph,
  char*filename,
  Boolean gen_ansi,
  char*msg
  )
```

filename は.XRT ファイルのパス名です。*gen_ansi* は、ANSI C コードを生成するかどうか指定します。エラーが起きると、`False` を返し、エラー・メッセージを *msg* に書きます。*msg* が `NULL` の場合は、エラーは *stderr* に書かれます。*msg* は最低 100 byte です。

XrtGraphInv.h ヘッダー・ファイルは必ずインクルードしてください。

3.4.1 コード生成プログラムで作成されたファイル

ソースは、ソース・コード生成プログラムが実行される度に上書きされますので、このソース・ファイルは、直接編集しないでください。

_xrt.c ファイル

_xrt.c ファイルは、グラフを作成したり、C コードで定義されたリソースをセットする手続きを定義します。

このファイルをアプリケーションにリンクして、グラフをアプリケーションにエンベッドすることができます。この手続きの名前は、グラフ名に依存します。例えば、.XRT ファイルの名が *mygraph.XRT* であれば、*mygraph_xrt.c* というソース・ファイルが次の手続きを定義します。

```
Widget
mygraph_xrt_create(parent)
Widget    parent;
{
/* Create graph & set resources etc. ... */
}
```

parent は、グラフを管理するウィジェットのハンドルです。*mygraph_xrt_create()*は、新しいグラフ・ウィジェットのハンドルを返します。

_xrt.res ファイル

_xrt.res ファイルは、グラフ・リソースのデフォルト値を持つ、Motif のリソースファイルです。[1] (プロパティ・エディタの Application Defaults を使用して)、リソースファイルでデフォルトと指定されているリソースだけが、このファイルで定義されます。その他のリソースは、*_xrt.c* ファイルで定義されます。

_xrt.h ファイル

_xrt.h ファイルは、*_xrt.c* ファイルの手続きを使用するソースコード・ファイルに必要な宣言を持っています。*_xrt_create()*ルーチンを呼ぶアプリケーションのソースコード・ファイルには、このファイルをインクルードしてください。

```
#include "mygraph_xrt.h"
```

_xrt.make ファイル

_xrt.make ファイルはサンプルの Makefile で、*make(1)*コマンドに、*xrt_gmf* で作成されたファイルからサンプル・アプリケーションをどのようにコンパイルするか指示します。このサンプル Makefile は、名前を変更してアプリケーションの Makefile のテンプレートとして使用することができます。

_xrt_main.c ファイル

_xrt_main.c ファイルは、サンプルの C プログラムで以下を実行します。

- Xt Intrinsics を初期化し、トップレベルのウィジェットを作成。
- データ・オブジェクトを作成し、*XrtDataCreateFromFile()*を使用してロードする。
- *_xrt.c* ファイルで定義されている*_xrt_create()*手続きを使用して、トップレベルで管理されるグラフを作成。
- データ・オブジェクトをグラフ・ウィジェットに接続する。
- *XtRealizeWidget()*と *XtMainLoop()*を呼び出す。

このサンプル・ソースは、名前を変更し、より複雑なアプリケーションの土台として使用できます。

---

[1] リソースファイルは、ウィジェットのリソース値の定義にフレキシブルな方法を提供します。例えば、このファイルを他のリソースファイルと結合させてアプリケーションのデフォルト・リソースファイルを作成することができます。アプリケーションのリソースファイルは、*/usr/lib/X11/app-defaults* にインストールできます。ユーザは、*.Xdefaults* ファイルでリソース値を変更することができます。尚、リソースファイルについての詳細は Xt Intrinsics に関する資料を参照してください。

3.4.2 ソースのコンパイルと実行

生成されたソースをコンパイルするには、"`make -f`"と入力し、その後に生成されたMakefileの名前を入力します。Makefileの名前が*mygraph_xrt.make*であれば次のように入力します。

```
make -f mygraph_xrt.make
```

ソース・ファイルが実行可能ファイルにコンパイルされる時、コンパイラとリンカの出力を確認できます。実行可能プログラムは、最後に"_xrt"が付きます(例:*mygraph_xrt*)。実行可能プログラムが作成されると、makefileは*xrt_auth*を呼び出し、XRT/graphのライセンスを調べ、そのプログラムに許可を与えます。コンパイルされたプログラムを実行するには、そのプログラム名を入力してください。

[Application Defaults]を用いて、*_xrt.res*ファイルにリソースが書かれるように指定した場合、これらのリソースは、アプリケーションが実行される前に、リソース・データベースにロードされている必要があります。次のように入力すると簡単に実行できます。

```
setenv XENVIRONMENT mygraph_xrt.res
```

4

XRT/graph のプログラミング

リソースについて ■ 軸のラベルと注釈法

軸のコントロール ■ 注釈のフォーマット

グラフ領域の位置決め ■ 3D 表示■ スタッキング

ヘッダー、フッター、凡例 ■ データ・スタイル

棒グラフの特別なリソース ■ 円グラフの特別なリソース

HiLo とキャンドル・グラフ ■ ポーラー/レーダー・グラフの特別なリソース

前景色と背景色 ■ マーカ

枠線 ■ 背景のイメージ ■ダブル・バッファリング

スクロールバー ■ 出力と印刷

4.1 リソースについて

この章では、XRT/graph のリソースについて解説します。これらのリソースは、覚えていただく必要はなく、ほとんどのグラフでは、リソースをデフォルト設定のままにしておくことができます。

Appendix A には XRT/graph の全てのリソースの一覧がありますので参照してください。

Resource	Type
XtNxrtData	XrtData *
XtNxrtData2	XrtData *
XtNxrtFrontDataset	int
XtNxrt[XYY2]Max	float
XtNxrt[XYY2]MaxUseDefault	Boolean
XtNxrt[XYY2]Min	float
XtNxrt[XYY2]MinUseDefault	Boolean

図 13　データのリソース

Resource	Type
XtNxrtDataAreaBackgroundColor	char*
XtNxrtDataAreaForegroundColor	char*
XtNxrtDataAreaImage	XrtImage
XtNxrtDataAreaImageLayout	XrtImageLayout(enum)
XtNxrtGraph3dShading	XrtShading(enum)
XtNxrtGraphBackgroundColor	char*
XtNxrtGraphBorder	XrtBorder(enum)
XtNxrtGraphBorderWidth	Dimension
XtNxrtGraphDepth	int
XtNxrtGraphForegroundColor	char*
XtNxrtGraphHeight	Dimension
XtNxrtGraphHeightUseDefault	Boolean
XtNxrtGraphImage	XrtImage
XtNxrtGraphImageLayout	XrtImageLayout(enum)
XtNxrtGraphInclination	int
XtNxrtGraphMarginBottom	Position
XtNxrtGraphMarginBottomUseDefault	Boolean
XtNxrtGraphMarginLeft	Position
XtNxrtGraphMarginLeftUseDefault	Boolean
XtNxrtGraphMarginRight	Position
XtNxrtGraphMarginRightUseDefault	Boolean
XtNxrtGraphMarginTop	Position
XtNxrtGraphMarginTopUseDefault	Boolean
XtNxrtGraphRotation	int
XtNxrtGraphShowOutlines	Boolean
XtNxrtGraphWidth	Dimension
XtNxrtGraphWidthUseDefault	Boolean
XtNxrtGraph[XY]	Position
XtNxrtGraph[XY]UseDefault	Boolean
XtNxrtType	XrtType(enum)
XtNxrtType2	XrtType(enum)
XtNxrtIsStacked	Boolean
XtNxrtIsStacked2	Boolean

図14　グラフのリソース

Resource	Type
XtNxrtPointLabels	char * *
XtNxrtPointLabels2	char * *
XtNxrtSetLabels	char * *
XtNxrtSetLabels2	char * *
XtNxrtTimeBase	time_t
XtNxrtTimeFormat	char *
XtNxrtTimeFormatUseDefault	Boolean
XtNxrtTimeUnit	XrtTimeUnit (enum)
XtNxrt[XYY2]AnnotationMethod	XrtAnnoMethod (enum)
XtNxrt[XYY2]AxisLabelFormat	char *
XtNxrt[XYY2]Labels	XrtValueLabel * *

図15　ラベルのリソース

Resource	Type
XtNxrtMarkerDataset	int
XtNxrtMarkerDataStyle	XrtDataStyle *
XtNxrtMarkerDataStyleUseDefault	Boolean
XtNxrt[XY]Marker	float
XtNxrtXMarkerMethod	XrtMarkerMethod (enum)
XtNxrtXMarkerSet	int
XtNxrtXMarkerPoint	int
XtNxrt[XY]MarkerDataStyle	XrtDataStyle *
XtNxrt[XY]MarkerDataStyleUseDefault	Boolean
XtNxrt[XY]MarkerShow	Boolean

図16　マーカのリソース

Resource	Type
XtNxrtDataStyles	XrtDataStyle * *
XtNxrtDataStyles2	XrtDataStyle * *
XtNxrtDataStylesUseDefault	Boolean
XtNxrtDataStyles2UseDefault	Boolean

図17　データ･スタイルのリソース

Resource	Type
XtNxrtAxisBoundingBox	Boolean
XtNxrtAxisFont	XmFontList
XtNxrtInvertOrientation	Boolean
XtNxrt[XYY2]AnnoPlacement	XrtAnnoPlacement(enum)
XtNxrt[XYY2]AnnotationRotation	XrtRotate(enum)
XtNxrt[XYY2]AxisMax	float
XtNxrt[XYY2]AxisMaxUseDefault	Boolean
XtNxrt[XYY2]AxisMin	float
XtNxrt[XYY2]AxisMinUseDefault	Boolean
XtNxrt[XYY2]AxisShow	Boolean
XtNxrt[XYY2]AxisLogarithmic	Boolean
XtNxrt[XYY2]AxisReversed	Boolean
XtNxrt[XYY2]Grid	float
XtNxrt[XYY2]GridUseDefault	Boolean
XtNxrt[XYY2]GridDataStyle	XrtDataStyle *
XtNxrt[XYY2]GridDataStyleUseDefault	Boolean
XtNxrt[XYY2]Num	float
XtNxrt[XYY2]NumUseDefault	Boolean
XtNxrt[XYY2]NumMethod	XrtNumMethod (enum)
XtNxrt[XYY2]Origin	float
XtNxrt[XYY2]OriginUseDefault	Boolean
XtNxrt[XYY2]OriginPlacement	XrtOriginPlacement (enum)
XtNxrt[XYY2]Precision	int
XtNxrt[XYY2]PrecisionUseDefault	Boolean
XtNxrt[XYY2]Title	char *
XtNxrt[XYY2]TitleRotation	XrtRotate (enum)
XtNxrt[XYY2]Tick	float
XtNxrt[XYY2]TickUseDefault	Boolean
XtNxrtYAxisConst	float
XtNxrtYAxisMult	float
XtNxrt[YY2]Axis100Percent	Boolean
XtNxrt[XYY2]AxisDataStyle	XrtDataStyle *
XtNxrt[XYY2]AxisDataStyleUseDefault	Boolean

図18　軸のリソース

Resource	Type
XtNxrtBarClusterOverlap	int
XtNxrtBarClusterWidth	int

図19　棒のリソース

Resource	Type
XtNxrtOtherDataStyle	XrtDataStyle *
XtNxrtOtherDataStyleUseDefault	Boolean
XtNxrtOtherLabel	char *
XtNxrtPieMergeMissingSlices	Boolean
XtNxrtPieMinSlices	int
XtNxrtPieOrder	XrtPieOrder (enum)
XtNxrtPieThresholdMethod	XrtPieThresholdMethod (enum)
XtNxrtPieThresholdValue	float
XtNxrtPieStartAngle	float

図20　円のリソース

Resource	Type
XtNxrtCandleComplex	Boolean
XtNxrtHiLoCloseShow	Boolean
XtNxrtHiLoOpenShow	Boolean
XtNxrtHiLoOpenCloseFullWidth	Boolean
XtNxrtBubbleMethod	XrtBubbleMethod (enum)
XtNxrtBubbleMin	float
XtNxrtBubbleMax	float

図21　HiLo/キャンドル/バブルのリソース

Resource	Type
XtNxrtAngleUnit	XrtAngleUnit (enum)
XtNxrtPolarHalfRange	Boolean
XtNxrtXOriginBase	float
XtNxrt[YY2]AnnotationAngle	float
XtNxrt[YY2]AnnotationAngleUseDefault	Boolean
XtNxrtPolarAxisAllowNegatives	Boolean

図22　ポーラー/レーダーのリソース

Resource	Type
XtNxrtHeaderAdjust	XrtAdjust (enum)
XtNxrtHeaderBackgroundColor	char *
XtNxrtHeaderBorder	XrtBorder (enum)
XtNxrtHeaderBorderWidth	Dimension
XtNxrtHeaderFont	XmFontList
XtNxrtHeaderForegroundColor	char *
XtNxrtHeaderHeight	Dimension
XtNxrtHeaderImage	XrtImage
XtNxrtHeaderImageLayout	XrtImageLayout (enum)
XtNxrtHeaderImageMinimumSize	Boolean
XtNxrtHeaderStrings	char * *
XtNxrtHeaderWidth	Dimension
XtNxrtHeader[XY]	Position
XtNxrtHeader[XY]UseDefault	Boolean
XtNxrtFooterAdjust	XrtAdjust (enum)
XtNxrtFooterBackgroundColor	char *
XtNxrtFooterBorder	XrtBorder (enum)
XtNxrtFooterBorderWidth	Dimension
XtNxrtFooterFont	XmFontList
XtNxrtFooterForegroundColor	char *
XtNxrtFooterHeight	Dimension
XtNxrtFooterImage	XrtImage
XtNxrtFooterImageLayout	XrtImageLayout (enum)
XtNxrtFooterImageMinimumSize	Boolean
XtNxrtFooterStrings	char * *
XtNxrtFooterWidth	Dimension
XtNxrtFooter[XY]	Position
XtNxrtFooter[XY]UseDefault	Boolean

図23　ヘッダーとフッターのリソース

Resource	Type
XtNxrtLegendAnchor	XrtAnchor (enum)
XtNxrtLegendBackgroundColor	char *
XtNxrtLegendBorder	XrtBorder (enum)
XtNxrtLegendBorderWidth	Dimension
XtNxrtLegendFont	XmFontList
XtNxrtLegendForegroundColor	char *
XtNxrtLegendHeight	Dimension
XtNxrtLegendImage	XrtImage
XtNxrtLegendImageLayout	XrtImageLayout (enum)
XtNxrtLegendOrientation	XrtAlign (enum)
XtNxrtLegendReversed	Boolean
XtNxrtLegendShow	Boolean
XtNxrtLegendWidth	Dimension
XtNxrtLegend[XY]	Position
XtNxrtLegend[XY]UseDefault	Boolean

図24　凡例のリソース

Resource	Type
XtNxrtExposeCallback	XtCallbackList
XtNxrtMapCallback	XtCallbackList
XtNxrtModifyCallback	XtCallbackList
XtNxrtPickCallback	XtCallbackList
XtNxrtResizeCallback	XtCallbackList
XtNxrtRotateCallback	XtCallbackList
XtNxrtTransformCallback	XtCallbackList
XtNxrtZoomAxisCallback	XtCallbackList

図25　コールバックのリソース

Resource	Type
XtNxrtBackgroundColor	char *
XtNxrtBorder	XrtBorder (enum)
XtNxrtDebug	Boolean
XtNxrtDoubleBuffer	Boolean
XtNxrtExtraDefaultDataStyles	Boolean
XtNxrtForegroundColor	char *
XtNxrtImage	XrtImage
XtNxrtImageLayout	XrtImageLayout (enum)
XtNxrtRepaint	Boolean

図26　その他のリソース

4.2 軸のラベルと注釈法

軸の注釈法や、凡例の各項目の横に表示されるテキストは、幾つかのリソースの組み合わせによって決定されます。軸の注釈法は、XtNxrt[XYY2]AnnotationMethod によって指定されます。有効な注釈法は次の通りです。

XRT_ANNO_VALUES (デフォルト)	グラフ・タイプとデータに基づいて、最も適する軸の注釈法が自動的に選択される
XRT_ANNO_POINT_LABELS (X 軸のみ)	X 値に基づいてポイントを X 軸上に配置し、指定したテキストで各ポイントに注釈を付ける
XRT_ANNO_VALUE_LABELS	任意の X または Y 座標に指定したテキストを使用して注釈を付ける
XRT_ANNO_TIME_LABELS (X 軸のみ)	X 値を時間の単位として認識し、指定された開始ポイントとフォーマットに基づいて、X 軸に時間ラベルで注釈を付ける

ポイント・ラベルと時間ラベルの注釈法は、X 軸でのみ使用できます。

軸の注釈法によって、使用されるリソースと無視されるリソースがあります。次のセクションからは、各注釈法をどのように使用するか解説します。

4.2.1 Values の注釈

XtNxrt[XYY2]AnnotationMethod が XRT_ANNO_VALUES の時、XRT/graph は幾つかのリソースを使用して、軸を注釈する数値をコントロールします。

数値の付け方

XtNxrt[XYY2]NumMethod は、軸の注釈法が XRT_ANNO_VALUES の時に、軸に付ける数値を指定します。有効な値は、次の通りです。

XRT_NUM_ROUND デフォルト	数値は、最小桁が 1、2、5 の倍数になるように四捨五入される。フォーマットは、XtNxrt[XYY2]Precision の値に従う。
XRT_NUM_PRECISION	数値は、XtNxrt[XYY2]Precision、XtNxrt[XYY2]Num と XtNxrt[XYY2]Tick の値によって計算される。

XRT_NUM_ROUND であれば、XRT/graph は、グラフ化されるデータを分析して、数値ラベルと目盛り(tick)の増分(increment)や数値の精度を決定します。次に、数値ラベルを四捨五入して軸を表示します。通常、この方法で最適な軸のラベルが得られます。

XRT_NUM_PRECISION であれば、数値は四捨五入されませんが、それ以外は、上記と同じように軸の目盛りが表示されます。

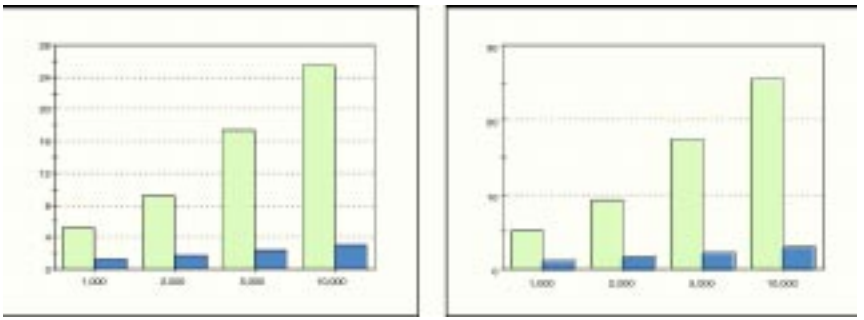


図 27 “ Precision"と"Rounded"を使用した軸のラベル

精度

XtNxrt[XYY2]Precision は、軸の注釈法によって働きが異なります。

- XtNxrt[XYY2]NumMethod が XRT_NUM_ROUND であれば、数値のフォーマットのみを指定します。
- XtNxrt[XYY2]NumMethod が XRT_NUM_PRECISION であれば、このリソースが、数値の表示を全体的にコントロールします。

XtNxrt[XYY2]Precision の正の値は小数点以下の桁数、負の値は小数点前の 0 の数を示します。

例えば、XtNxrtXPrecision は-3 で XtNxrtYPrecision が 2 の場合、X 軸の数値は 1000 の倍数で、Y 軸の数値は小数第二位まで表示されます。

XtNxrt[XYY2]PrecisionUseDefault が True であれば、XRT/graph が実行時に最適な数値の精度を決定します。

数値の増分(increment)

XtNxrt[XYY2]Num リソースは、軸の数値ラベルの間隔を指定します。例えば、XtNxrtYNum が 3.2 であれば、XRT/graph は原点から 3.2 単位毎に Y 軸に数値ラベルを付けます。

軸に表示される目盛り(tick)は、数値ラベルの増分に関係します。目盛りに関する解説は、4.3.3 にありますので参照してください。

数値ラベルや目盛、グリッド線のリソースを小さな値に設定すると、グラフが表示された時に読みにくくなります。XRT/graph に、最適な数値ラベルと目盛りの増分を計算させるには、UseDefault を True に設定してください。

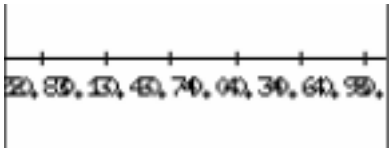


図28　数値ラベルの増分の設定が小さすぎる例

注釈のフォーマット

軸の注釈は、XtNxrt[XYY2]AxisLabelFormat リソースを使用して、会計、通貨、パーセンテージ、指数等のフォーマットが使用できます。詳細は、4.4 を参照してください。

対数(logarithmic)の軸

linear から log10 の軸に変換するには、XtNxrt[XYY2]AxisLogarithmic を True にします。対数軸は、0 より大きい値のみをグラフ化するので、0 以下のデータは hole(欠けているデータ値)とみなされます。対数軸には、数値ラベルや目盛りの増分、精度は指定できません。また、軸、データの最小/最大、原点は、0 以下に設定することはできません。

対数軸には、次の制限があります。

- グラフ・タイプは、線、面、HiLo、HiLoOpenClose、キャンドルのみ。
- X 軸の注釈法は、XRT_ANNO_TIME_LABELS にセットできない。

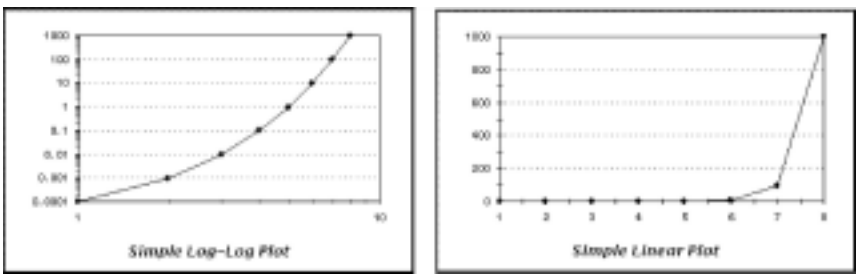


図29　Linear と Logarithmic の軸

4.2.2　ポイント・ラベルの注釈とセット・ラベル

XtNxrtXAnnotationMethod が XRT_ANNO_POINT_LABELS の時、XRT/graph は、ポイント・ラベルとセット・ラベルを使用して配列(Array)データの行と列にラベルを付けます。ポイント・ラベルが X 軸の注釈に、セット・ラベルが凡例に使用されます。データが一般型(General)なら、セット・ラベルがデータのラベル付けに使用されます。[1]

[1] 一般(General)データの場合、各セットは、それぞれ異なる X 値とポイント数を持つので、ポイント・ラベルは意味がありません。

図30は、*XrtDataCreateFromFile()*でロードする場合に適したフォーマットを持つ *pop.dat* ファイルです。これは 3 都市の過去と未来の推定の人口のデータで、(3、5)の配列を定義しています。

```
Figure 30 The pop.dat File
ARRAY 3 5
# X Samples (One for the beginning of each decade 1960 to 2000)
1 2 3 4 5
# Mexico
5 8 14 19 24
# Tokyo
11 15 18 21 22
# New York
14 16 15.5 15.2 15.4
```

図30　pop.dat ファイル

XRT/graphでは、このファイルは、「5ポイントが3セットある」といいます。セット・ラベルは都市名、ポイント・ラベルは年です。このファイルが、popと呼ばれるデータ・オブジェクトの作成に使用される時、次のコードでラベルを定義できます。(その結果のグラフは図31です。)

```
static char *sl[] = { "Mexico City", "Tokyo",
        "New York", NULL };
static char *pl[] = { "60", "70", "80", "90",
        "2000", NULL };
popg = XtVaCreateManagedWidget("graph",
  xtXrtGraphWidgetClass, parent_widget,
  XtNxrtXAnnotationMethod, XRT_ANNO_POINT_LABELS,
  XtNxrtData,      pop,
  XtNxrtSetLabels  sl,
  XtNxrtPointLabels,    pl,
  NULL);
```

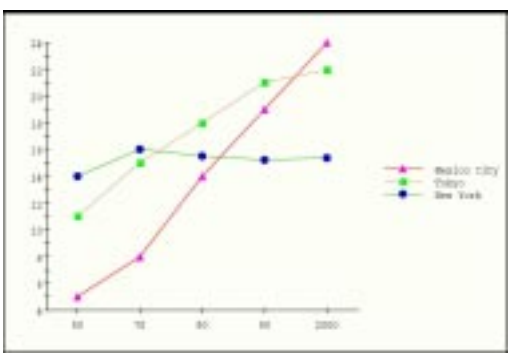


図31　ポイント・ラベルのX軸と、セット・ラベルの凡例

XtNxrtSetLabels と XtNxrtPointLabels を使用する代わりに、*SetNth* と *GetNth*(Appendix D を参照)を使用して、個々のラベルにアクセスできます。

```
/* Note that index is zero-based */
printf("Third Set-label is %s\n",
    XrtGetNthSetLabel(graph, 2) );

/* Make 3rd Point-label "eighty" */
XrtSetNthPointLabel(graph, 2, "eighty");
```

グラフのデータが XRT_DATA_GENERAL の場合は、ポイント・ラベルは無視されます。

4.2.3 値ラベルの注釈

XtNxrt[XYY2]AnnotationMethod が XRT_ANNO_VALUE_LABELS の時、XRT/graph は、指定されたテキスト・ストリングを軸のラベルとして使用します。各値ラベルは、XrtValueLabel 構造体で定義されます。

```
typedef struct {
  float   value;
  char*string;
} XrtValueLabel;
```

value は軸のラベルが表示される X、Y、Y2 値を定義し、*string* は表示されるテキストです。値ラベルの配列全体は、XtNxrt[XYY2]Labels を用いてプログラミングできます。また、*XrtGetValueLabel()*や *XrtSetValueLabel()*で個々の値ラベルにアクセスすることもできます。

値ラベルは、X 軸や Y 軸に重要な値を表示する時や、任意の軸の注釈が必要な場合に使用します。例えば、X 軸の(X=7.43)に、"foobar"という値ラベルを付ける時には、次のコードが使用できます。

```
XrtValueLabel mylabel;
my_label.value = 7.43;
my_label.string = "foobar";
XrtSetValueLabel(graph, XRT_AXIS_X, &mylabel);
```

*XrtGetValueLabel()*は XrtValueLabel 構造体のポインタを取り、他の XrtValueLabel 構造体のポインタを返します。返された構造体は、渡された構造体の値に一番近い値ラベルの値とストリングのペアでセットされます。値ラベルがなければ、NULL を返します。次の例は、X=5.0 に一番近い値ラベルを捜しています。

```
XrtValueLabel my_label, *closest_label;

my_label.value = 5.0;
closest_label = XrtGetValueLabel(graph, XRT_AXIS_X, &my_label);
if (closest_label != NULL) { /* all parameters valid */
    if (closest_label->string == NULL)
      printf("No X-axis Value-labels exist!\n");
    else
      printf("Closest is at %f, text is %s\n",
        closest_label->value,
        closest_label->string);
}
```

軸の値ラベルを全てクリアするには、XtNxrt[XYY2]Labels を NULL にセットしてください。

特定な値ラベルを削除する場合は、その *string* を NULL にします。値ラベルを削除する時は、浮動小数点の四捨五入の際の誤差に注意してください。値ラベルの削除では、最初にそれを取得し、そのストリングを NULL に設定、そしてまたセットする方法が最適です。次の例では、X=5.0 に一番近いラベルを削除しています。

```
XrtValueLabel label, *label_ptr;

label.value = 5.0;
label_ptr = XrtGetValueLabel(graph, XRT_AXIS_X, &label);
if (label_ptr != NULL) { /* all parameters valid */
    if (label_ptr->string == NULL)
      printf("No labels to delete!\n");
    else {
      label = *label_ptr;
      label.string = NULL;
      XrtSetValueLabel(graph, XRT_AXIS_X, &label);
    }
```

値ラベルを使用して、X 軸に特定な日付を付けたり、Y 軸に値の代わりにテキストを付けたりすることができます。図32 は、両方の軸に値ラベルを使用しています。

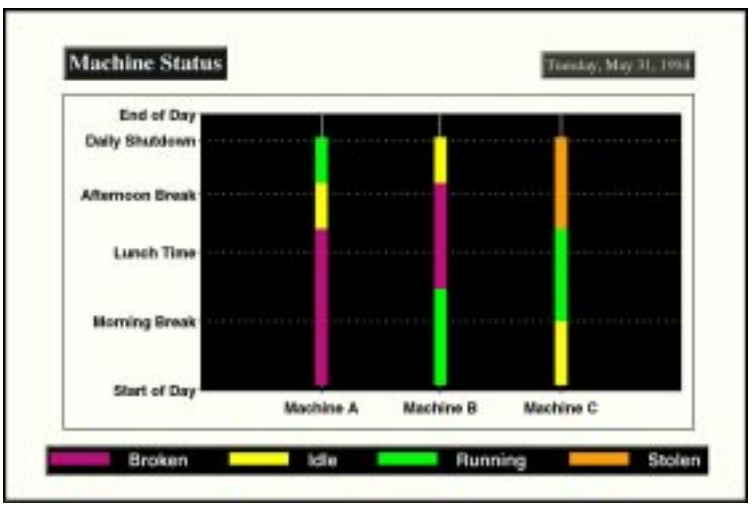


図32　値ラベルを使用した例

4.2.4 時間軸のラベル

XRT/graph は、時間の次元を表現する X 軸をサポートしています。時間軸を指定し、コントロールするリソースは以下の通りです。

XtNxrtTimeBase	X 値が測定されはじめる時間を指定
XtNxrtTimeUnit	X 値が測定される時間の単位を指定
XtNxrtTimeFormat	時間軸の注釈のフォーマットを指定
XtNxrtTimeFormatUseDefault	時間のフォーマットがダイナミックに計算されるよう指定
XtNxrtXAnnotationMethod	X 軸の注釈法を指定、時間軸の場合は、XRT_ANNO_TIME_LABELS に設定

時間軸の規準

時間軸は、時間の単位(秒、分、時、日、週、月、年)で測定されるグラフのアプリケーションに適しています。時間軸の指定には、幾つかの規準があります。

- グラフ・タイプは、円、ポーラー、レーダー、filled レーダー以外であること
- 軸で表現される年が 1970 年以降、2038 年以前であること
- 注釈の単位が秒より小さくないこと

もし、上記の条件にあわない場合は、値ラベル等の他の方法を使用してください。

時間の単位

XtNxrtXAnnotationMethod を XRT_ANNO_TIME_LABELS に設定すると、XRT/graph はデータの X 値を時間の単位を使用して解釈します。秒がデフォルトの単位ですので、X 値の範囲が 20 から 55 であれば、XRT/graph は X 軸を 35 秒の長さにします。[1]

XtNxrtTimeUnit を使用して、時間の単位を設定できます。設定できる値は次の通りです。XRT_TMUNIT_SECONDS、XRT_TMUNIT_MINUTES、XRT_TMUNIT_HOURS、XRT_TMUNIT_DAYS、XRT_TMUNIT_WEEKS、XRT_TMNIT_MONTHS、XRT_TMUNIT_YEARS

基準時間

データの X 値は、基準時間をもとに測定されます。例えば、X 値が 1993 年の日であれば(9 は 1 月 10 日、31 は 2 月 1 日、etc)、基準時間を 1993 年の始めに設定してください。XtNxrtTimeBase は、X 値の測定が開始される時間を定義します。

[1] 但し、XtNxrtXAxisMin と XtNxrtXAxisMax、XtNxrtXMin と XtNxrtXMax の値を設定していない場合。

このリソースは、ダイナミックにデフォルトが1970年1月1日にセットされる *long* で、1970年1月1日GMT(グリニッジ標準時)からの時間を秒で表します。通常、データはこれ以外の基準を使用することが多いので、時間軸を表示する時は、必ずこのリソースを設定してください。

*XrtMakeTime()*または、ANSIの *mktime()*は、日時を1970年1月1日からの秒に変換する時に便利です。例えば、データの基準時間が1993年4月19日08:30:05であれば、次のように設定します。

```
long tval;
struct tm ltime;

/* Set time base to April 19, 1993, 08:30:05 */
tval = XrtMakeTime(93, 4-1, 19, 8, 30, 5);
XtVaSetValues(graph,
  XtNxrtTimeBase, tval, NULL);
```

時間のフォーマット

XRT/graphは、`XtNxrtTimeFormat`で指定されたストリング値と、使用しているシステムのANSI Cの *strftime()*を時間軸の注釈のフォーマットに使用します。[1] *strftime()*はシステムによって異なる事がありますので、使用しているシステムの資料でどのようにこのストリングが解釈されるか調べてください。Appendix Aには、`XtNxrtTimeFormat`の解説がありますので参照してください。

XRT/graphにデフォルトで最適な時間のフォーマットを計算させるには、`XtNxrtTimeFormatUseDefault`を`True`に設定してください。

次の表は、1993年4月12日月曜日、06:55:50がどのようにフォーマットされるかの例です。

Time Format String	例
%Y	1993
%b %y	Apr 93
%b	Apr
%b %d	Apr 12
%a %d	Mon 12
%a	Mon
%a %H:00	Mon 06:00
%H:%M	06:55
%H:%M:%S	06:55:50

時間の軸に目盛(tick)を表示するには、`XtNxrtXTickUseDefault`を`True`に設定してください。

[1] *strftime()*はロケールの国際化標準を使用しており、時間軸の注釈は *locale* でサポートされる言語と環境に自動的に対応します。システムによっては、1969年より前の値を無効にしてしまうことがあるので、基準時間を1970年1月1日より前の値には設定しないでください。

時間の精度

XRT/graph で表示されるデータは、単精度の浮動小数点値で表されます。ほとんどの機種は *float* に IEEE(6 桁の精度)を使用しているので、XRT/graph は ANSI や POSIX(4 byte *long* を使用)のように多くの時間の値を表現することはできません。次の表は時間の値の浮動小数点表示固有の誤差を示しています。

時間	秒	誤差
1 日	86,400	＜1 秒
1 年	31,536,000	＜2 分
1970 年から	～730 million	＜17 分

この表が示すように、一年間での"時間"の単位は正確ですが、"分"は不正確です。時間の単位や `XtNxrtTimeBase` を指定する時には、この四捨五入の誤差を考慮して設定してください。

例えば、一年間分のデータがあり、X 値はその年の始めから秒で測定されるとします。`XtNxrtTimeBase` をその年の始めに設定すると、その一年間のデータは時(hour)の単位の精度までなら正しく表示することができますが、それ以上細かい単位で表示しようとすると、1、2 分の誤差が生じます。もし、細かい単位が必要な場合は、一日で表現されるデータに切り替え、`XtNxrtTimeBase` をその日の始めに設定することにより、秒の単位まで正確に表示できます。

時間軸の例

time.c(*$XRTHOME/src/graph/examples*)は、*data/time.dat* からデータ・ファイルを読み込みます。このデータ・ファイルは 12(0 から 11)の X 値を持ち、1992 年の 12 ヶ月を表しています。*time.c* がコンパイル、実行されると、図 33 が表示されます。

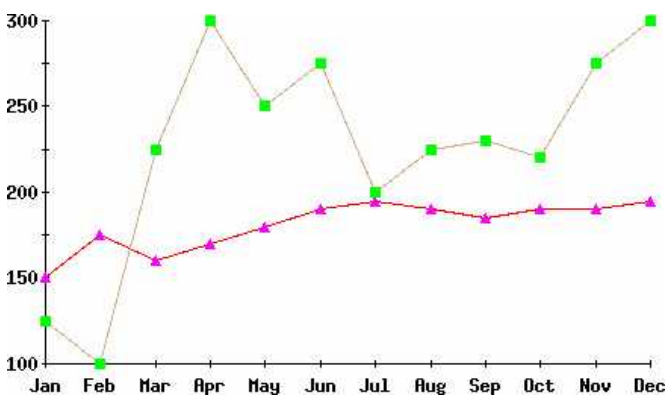


図 33　*time.c* によって作成されたグラフ

```
Figure 34 The data/time.dat data file
ARRAY 2 12
# x values, for jan to dec 1992.
0 1 2 3 4 5 6 7 8 9 10 11
# y values line 1
150 175 160 170 180 190 195 190 185 190 190 195
# y values line 2
125 100 225 300 250 275 200 225 230 220 275 300
```

図34　data/time.data データ・ファイル

```
#include <time.h>
#include <Xm/XrtGraph.h>

main(int argc, char **argv)
{
    XtAppContext app_context;
    Widget graph, toplevel;
    XrtDataHandle my_data;
    time_t tval;
    static struct tm ltime;

    /* Allocate and load the data */
    my_data = XrtDataCreateFromFile("data/time.dat", NULL);

    /* Calculate basetime of Jan 1, 1992 */
    ltime.tm_year = 92;
    ltime.tm_mday = 1;
    time.tm_isdst = -1; /* mktime figures out TZ effect */
    tval = mktime(&ltime);

    /* Initialize Intrinsics */
    toplevel = XtVaAppInitialize(&app_context, "Time", NULL, 0, &argc, argv,
          NULL, NULL);
    /* Create graph */
    graph = XtVaCreateManagedWidget("mygraph", xtXrtGraphWidgetClass, toplevel,
      XtNxrtData,      my_data,
      XtNxrtTimeBase,       tval,
      XtNxrtTimeUnit,       XRT_TMUNIT_MONTHS,
      XtNxrtXAnnotationMethod,  XRT_ANNO_TIME_LABELS,
      XtNwidth,             500,
      XtNheight,            300,
      0);
    XtRealizeWidget(toplevel);
    XtAppMainLoop(app_context);
}
```

図35　time.c プログラム

4.3 軸のコントロール

このセクションでは、X 軸と Y 軸の表示のコントロール法を解説しています。Y2 軸は、7.1 に解説されています。

4.3.1 軸とデータの範囲

デフォルトでは、XRT/graph は、データ・オブジェクトのデータを全て表示します。XRT/graph は、データ値の最小と最大(XtNxrt[XYY2]Min と XtNxrt[XYY2]Max）、原点、軸の目盛りの増分に基づいて、軸の最小と最大 (XtNxrt[XYY2]AxisMin と XtNxrt[XYY2]AxisMax）を決定します。グラフ化されるデータの範囲は、軸またはデータの最小と最大を指定することにより変更することができます。XRT/graph では、グラフ・データの範囲の設定には、データを指定するより軸を指定する方が、良い結果が得られます。

軸の最小と最大

XtNxrt[XYY2]AxisMin と XtNxrt[XYY2]AxisMax は、表示される軸の最小値と最大値を指定します。特定な軸の値でデータの範囲を設定する場合には、これらのリソースを使用してください。次の例は、グラフの中央を中心に 2 倍にズームしています。

```
float xmax, xmin, ymax, ymin;
float xmax2, xmin2, ymax2, ymin2;
XtVaGetValues(graph,
  XtNxrtXAxisMin, &xmin,
  XtNxrtXAxisMax, &xmax,
  XtNxrtYAxisMin, &ymin,
  XtNxrtYAxisMax, &ymax,
  NULL);
xmin2 = xmin + (xmax - xmin) / 4.0;
xmax2 = xmax - (xmax - xmin) / 4.0;
ymin2 = ymin + (ymax - ymin) / 4.0;
ymax2 = ymax - (ymax - ymin) / 4.0;
XtVaSetValues(graph,
  XtNxrtYAxisMin, XrtFloatToArgVal(xmin2),
  XtNxrtXAxisMax, XrtFloatToArgVal(xmax2),
  XtNxrtYAxisMin, XrtFloatToArgVal(ymin2),
  XtNxrtYAxisMax, XrtFloatToArgVal(ymax2),
  NULL);
```

XRT/graph は、ハードコーディングされている無効な軸の範囲を、対応する UseDefault を True に設定して置き換えます。例えば、XtNxrtYAxisMax が -20(XtNxrtYAxisMaxUseDefault は False）で、XtNxrtYAxisLogarithmic が True に設定されると、対数軸は 0 より大きい値だけをグラフ化するので、XtNxrtYAxisMaxUseDefault は True になります。

データの最小と最大

XtNxrt[XYY2]Max と XtNxrt[XYY2]Min は、表示されるデータの最大値と最小値を設定します。特定な *data* 値で、グラフ化されるデータの範囲を指定する時は、これらのリソースを使用してください。(これらのリソースに対してユーザが指定した値は、グラフの種類によっては無視されることがあります。)

4.3.2 原点

原点座標

原点の座標は XtNxrt[XYY2]Origin で指定します。Y 軸を X=10 で X 軸と交差させるには、XtNxrtXOrigin を 10.0 に設定します。

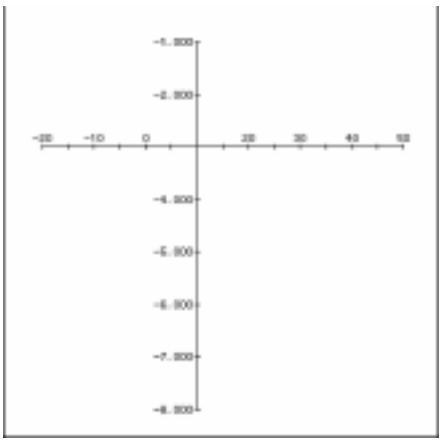


図 36　原点(10、-3)、X 精度(-1)、Y 精度(3)

原点の位置

XtNxrt[XYY2]OriginPlacement により、軸の原点の位置を設定します。有効な値は次の通りです。

XRT_ORIGIN_AUTO デフォルト	線と面グラフでは、原点は軸の最小値または(データセットに正と負の値がある時は)0 に配置される。棒、積み上げ棒、HiLo、HiLoOpenClose、キャンドルでは、原点は 0 に配置される。
XRT_ORIGIN_ZERO	原点は、0 に配置される。
XRT_ORIGIN_MIN	原点は、軸の最小値に配置される。
XRT_ORIGIN_MAX	原点は、軸の最大値に配置される。

このリソースは、XtNxrt[XYY2]OriginUseDefault が False の時は、無視されます。XtNxrt[XYY2]Origin によって原点の座標が明示的に設定されている場合は、このリソースは上書きされます。

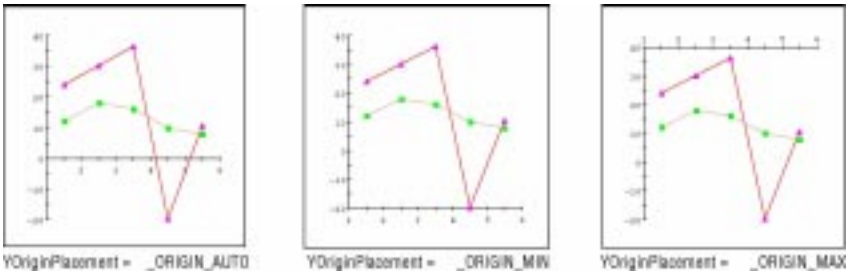


図37　原点の位置

4.3.3　目盛りとグリッド線の増分(increment)

目盛り(tick)の増分

軸に付けられる目盛り(tick)の増分は、XtNxrt[XYY2]Tick によってコントロールされます。目盛りが数値ラベルの代わりに付けられる事を除けば、XtNxrt[XYY2]Tick は XtNxrt[XYY2]Num と似ています。数値ラベルの増分は、必ずしも目盛りの増分の倍数である必要はありません。

グリッド線

XtNxrt[XYY2]Grid は、グリッド線の間隔を指定します。デフォルトでは、このリソースは 0.0 で、グリッド線は表示されません。各軸ラベル上でグリッド線を表示するには、XtNxrt[XYY2]GridUseDefault を True にセットしてください。任意の増分を指定する場合は、XtNxrt[XYY2]Grid を使用します。

グリッド線のスタイルは、XtNxrt[XYY2]GridDataStyle と XtNxrt[XYY2]GridDataStyleUseDefault を使用して指定できます。

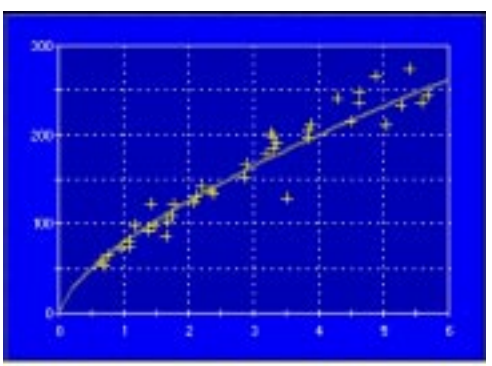


図38　グリッド線と軸の目盛り

4.3.4 軸のタイトルと注釈の回転

タイトル

XtNxrt[XYY2]Title を使用して、各軸のタイトルを指定することができます。但し、ポーラー、レーダー、filled レーダーでは、このリソースは使用できません。

タイトルの回転

縦軸(XtNxrtInvertOrientation が True でなければ、Y と Y2 軸）のタイトルは、XtNxrt[XYY2]TitleRotation を XRT_ROTATE_90 に設定することにより、時計の針と反対方向に 90 度回転します。

このリソースを XRT_ROTATE_270 に設定すると、タイトルは時計の針と反対方向に 270 度回転します。Y2 軸のタイトルは、通常、この角度が適しています。角度を水平に戻すには、このリソースを XRT_ROTATE_NONE に設定してください。

横軸(XtNxrtInvertOrientation が True でなければ、X 軸）のタイトルは回転できません。

注釈の回転

軸の注釈は、XtNxrt[XYY2]AnnotationRotation を使用して回転できます。但し、極、レーダ、filled レーダでは、X 軸の回転は無視されます。

回転したテキストは、グラフのリアルタイムのパフォーマンスを落とします。(速度が 10 倍程度落ちることもあります。)リアルタイムのパフォーマンスが重要なアプリケーションでは、回転したテキストを使用しないでください。

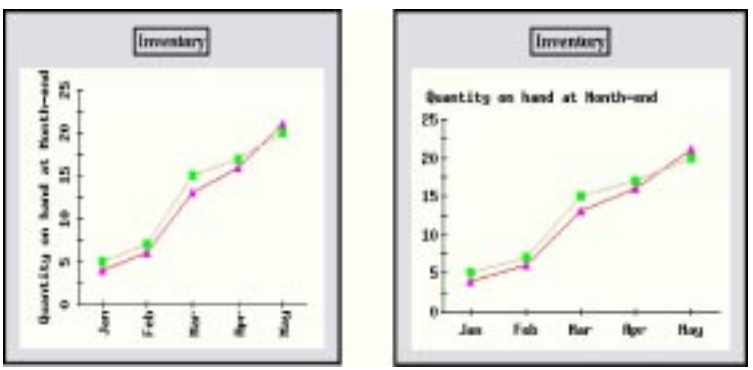


図 39　90 度に回転させた注釈とタイトル

4.3.5 その他の軸のコントロール

方向

XtNxrtInvertOrientation が False の時は、全ての X 軸のリソースは横軸に、Y 軸のリソースは縦軸に適用されます。XtNxrtInvertOrientation が True であれば、X 軸は縦軸、Y 軸は横軸になります。

マージン

グラフ・マージンのリソース(XtNxrtGraphMarginTop、XtNxrtGraphMarginBottom、XtNxrtGraphMarginLeft、XtNxrtGraphMarginRight)を用いてグラフ領域の端と軸とのスペースを調節できます。[1] 各リソースは、グラフの端と軸との間のピクセル数を指定します。デフォルトでは、XRT/graph は、グラフ・マージンを自動的に計算して(UseDefault リソースが True)、軸の注釈やタイトルが表示されるよう十分なスペースを設定します。

グラフ・マージンは、次のような時に使用します。

- 複数のグラフ・ウィジェットの整列。例えば、XtNxrtGraphMarginLeft を用いて、幾つかのグラフを Y 軸に沿って並べることができます。
- グラフのスケーリング。グラフ・マージンのリソースを負の値にセットして"ズームイン"、正の値にセットして"ズームアウト"できます。
- グラフの移動。例えば、XtNxrtGraphMarginTop を増加し、XtNxrtGraphMarginBottom を減少させて、グラフを下の方向に移動できます。



図 40　マージンを使用したグラフのスケーリング

軸のフォント

デフォルトでは、軸の数値ラベルとタイトルは、fixed フォントを使用しています。XtNxrtAxisFont を用いて、現在のロケールの設定でサーバがサポートしているフォントなら、どれでも設定できます。軸のフォントの変更は、ヘッダー、フッター、凡例の場合(4.8 を参照)と同様です。

軸の反転

X 軸、Y 軸、Y2 軸は、XtNxrt[XYY2]AxisReversed を True に設定することにより、方向を変換できます。

Y 軸と Y2 軸の関係が一度設定されると、Y 軸の方向だけしか指定できません。もし、関係を表す倍数が負の数なら、Y2 軸は Y 軸と反対方向になります。

[1] グラフ・マージンは、エンドユーザがインタラクティブに調節することもできます。インタラクティブなスケーリング、移動、ズームについての詳細は、6.1 に解説されています。

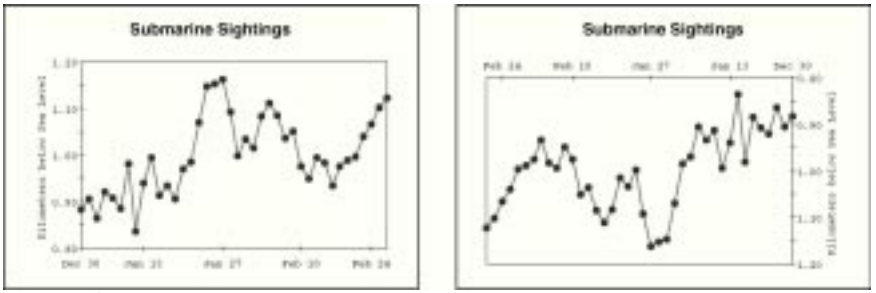


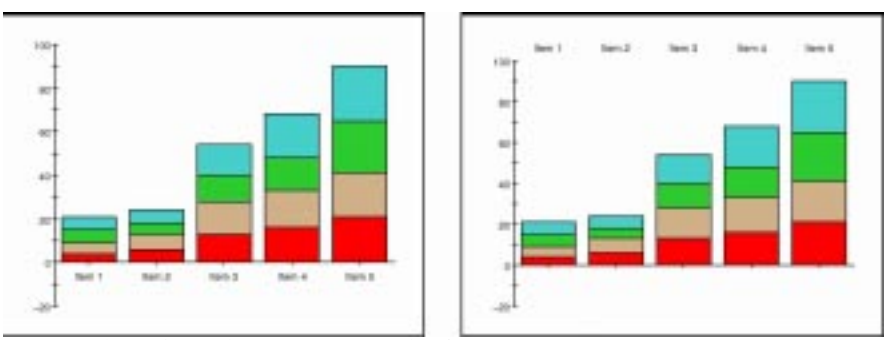
図41　普通の軸と反転した軸

注釈とタイトルの位置

XtNxrt[XYY2]AnnoPlacement を使用して、軸の注釈とタイトルの位置をコントロールすることができます。各軸の注釈/タイトルは、原点、軸の最小/最大に表示させることができます。有効な値は、次の通りです。

XRT_ANNO_AUTO (デフォルト)	注釈は自動的に配置される。線と面グラフでは、注釈は原点に配置され、その他のグラフでは、注釈は、軸の終点で原点に近い方に配置される。
XRT_ANNO_ORIGIN	注釈は原点に配置される。
XRT_ANNO_MIN	注釈は軸の最小に配置される。
XRT_ANNO_MAX	注釈は軸の最大に配置される。

二本の軸のグラフでは、Y 軸の注釈の位置の指定は無視されます。また、Y2 軸の注釈の位置は、Y2 軸が primary(第一の)Y 軸になっていなければ、無視されます。


図42　Y 軸の原点と最大に配置された X 軸の注釈

軸の表示

XtNxrt[XYY2]AxisShow は、XRT/graph に軸の表示の有無を指示します。False であれば、軸は表示されません。

枠

軸を囲むボックスを描く時は、XtNxrtAxisBoundingBox を True にセットしてください。

軸のデータ･スタイル

軸の色、幅やパターンなどは、XtNxrt[XYY2]AxisDataStyle でコントロールできます。詳細は、4.9 を参照してください。

4.4 注釈のフォーマット

軸の注釈法が Values(XtNxrt[XYY2]AnnotationMethod が XRT_ANNO_VALUES)の時、数値ラベルを様々なフォーマットで表示することができます。XtNxrt[XYY2]AxisLabelFormat を使用して、軸の注釈のフォーマット(フォーマット･コードのストリング)を各軸ラベルに指定してください。図 43 は、それぞれのフォーマットを使用した例です。次のコードは、Y 軸のフォーマットを my_format にセットしています。

```
char * my_format = “$#,##0.00;-$#,##0.00”;
XtVaSetValues(graph,
  XtNxrtYAxisLabelFormat, my_format,
  NULL);
```

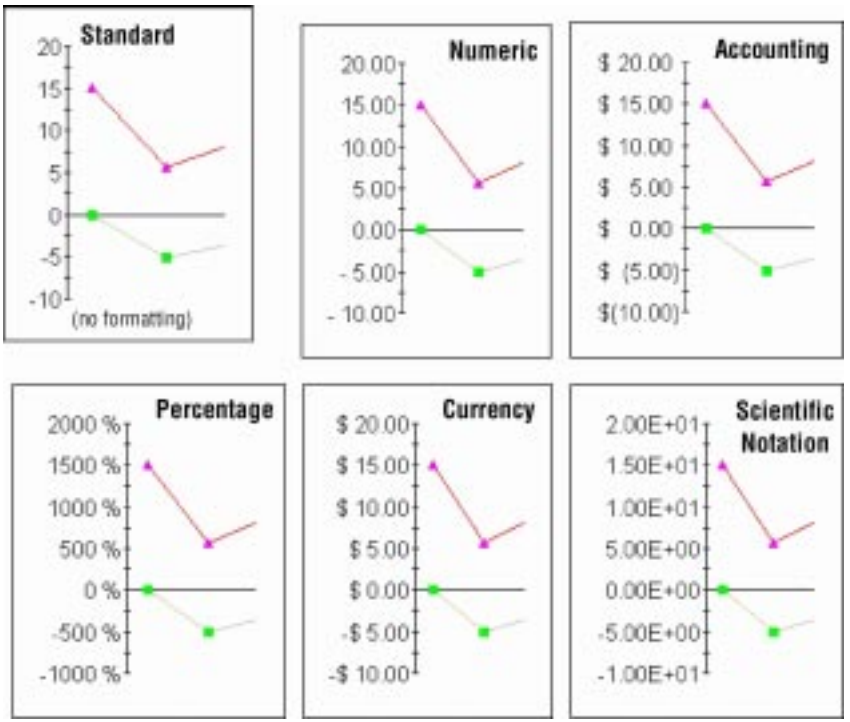


図43　軸のフォーマット例

フォーマットのコマンド

フォーマットは、”#”、”0”、”$”、”.”、”[multiplier 0.1]”等のコマンドで指定します。フォーマット・コマンドは、”;”(セミコロン)で区切ってグループ分けし、正の値、ゼロ、負の値に異なるフォーマットを定義することができます。最初のグループは、常に正の値をフォーマットします。グループが二つ以下の場合は 0 の値、グループが一つしかない時は負の値も、この最初のグループでフォーマットされます。二番目のグループは常に負の値に、三番目のグループは常に 0 の値のフォーマットに使用されます。特別な範囲の数値には、それ以上の数のグループを指定することもできます。これは、このセクションの後半で解説されています。

一般的なフォーマット指定

次の表は、一般的なフォーマットの指定です。

	フォーマットの指定	フォーマットされた注釈の例
標準	NULL	-20　0　　140　1200
数値(桁数指定、四捨五入される)	0.00;-0.00 #,##0.0	-20.00　0.00　140.00　1200.00 -20.0　0.0　140.0　1,200.0
通貨	$#,##0.00;-$#,##0.00 $#,##0.00;$(#,##0.00)	-$20.00　$0.00　$140.00　$1,200.00 $(20.00)　$0.00　$140.00　$1,200.00

会計(小数点と通貨シンボルを行揃え)	_-$* #,##0.00;-$* #,##0.00 $* _(#,##0.00_);$* (#,##0.00)	-$20.00 $0.00 $140.00 $1,200.00 $(20.00) $0.00 $140.00 $1,200.00
パーセンテージ (100 を乗算、%を追加)	0.00%;-0.00% 0 %;-0 % %0;%-0	-2000.00% 0.00% 14000.00% -2000% 0% 14000% %-2000 %0 %14000
分数	0 ?/? 0 ?/8	-5 1/4 0 140 3/5 -5 2/8 0 1200 7/8
指数	0.00E+00 0.00e-00 0.0[hide]E[show]+00	-2.00E+01 0.00E+00 4.00E+01 -2.00e01 0.00e00 4.00e01 -2.0+01 0.0+00 4.0+01

フォーマット・コード

フォーマットの指定に使用できるフォーマット・コードの一覧です。

`# 0 ?` 数値が入る場所を指定。 .(ピリオド)と一緒に使用して表示桁数を指定。桁数が#で指定された桁に満たない時は 0 は挿入されない。0 は、指定された桁数より実際の桁が少ない時に、その桁に 0 を表示する。?は、0 と同様の働きをするが、小数点や分数を行揃えするために、指定された桁位置に該当する値がない部分にスペースを挿入する。例えば、“#.0#”は、16 を 16.0 に、98.666 を 98.67 にフォーマットする。

`. (ピリオド)` 桁数と四捨五入を指定する#、0、?と一緒に使用し、小数点の位置を指定する。例えば、”###.0000”は、123.45 を 123.4500 にフォーマットする。

`, (カンマ)` 桁と桁の間にある場合は、1000 毎の桁区切りとなる。値の最後にある場合は、値を 1000 で割る。例えば、”#,###”は、21000 を 21,000 に、”0.0,”は、1200 を 1.2 にフォーマットする。

`$ - + / ( )` 指定場所に、これらの文字を挿入する。例えば、”$#.00”は、23.3 を$23.30 にフォーマットする。

`\\<character>` 指定場所に、<character>の文字を挿入する。例えば、”\\%”は、(数値を 100 倍せずに)%を挿入する。

`_<character>` 指定場所に、<character>の文字幅と同じスペースを挿入する。例えば、”_-“は、マイナス・サインと同じ幅のスペースを挿入する。

`%` 数値を 100 倍し、%を挿入する。例えば、”#.##%”は、.85 を 85%にフォーマットする。

`[multiplier <n>]` 次に続く値を<n>倍する。例えば、”[multiplier 1.6]は、10 を 16 にフォーマットする。

E+ e+ E- e-	指数の前にEまたはeを挿入して指数で表記する。E+とe+は、指数が正の場合は+(プラス記号)、指数が負の場合は-(マイナス記号)を指数の前に付ける。E-とe-は、指数が負の場合にのみ-を付ける。例えば、"0.0E+00"は、18987 を 1.8E+04 にフォーマットする。
*<character>	注釈に適当なスペースを挿入する。例えば、"$*##.##"は、$と値の間に、充分なスペースを挿入する。
[hide] [show]	[hide]は、その後に続く値またはフォーマットを非表示にする。[show]は、値/フォーマットの非表示を表示に戻す。例えば、"0.0[hide]E[show]+00"は、70 を 7.0+01 とフォーマットする。
: (セミコロン)	フォーマットのグループを区切る。(下の項の例参照)

特別な範囲のフォーマット

デフォルトの正の値、負の値、0 のグループ以外に、特別な数値の範囲に対しても、フォーマットを定義することができます。[>value]、[<value]、[=value]、[< >value]、[>=value]、[<=value]を使用して、新しいグループを定義し、この条件に合う数値にだけ適用されるフォーマット・コードを指定することができます。各グループは(;)で区切られます。次の例は、1000 より大きい値に特別なフォーマットを定義しています。

```
###.##;-###.##;#.#;[>1000]0.0E+00
```

フォーマット指定の検査

最も適するフォーマット・コードの組み合わせを決定する事は、時間がかかる場合があります。*XrtCustomFormatValidate()* を呼び出すことにより、フォーマットを XtNxrt[XYY2]AxisLabelFormat にセットする前に、チェックすることができます。この手続きは、Appendix D に解説されています。

4.5 グラフ領域の位置決め

ヘッダー、フッター、凡例、グラフの各領域は、XRT/graph によって実行時にデフォルトの位置に配置されます。各領域のデフォルトの位置決めは、以下の内容を始め、あらゆる要因に依存します。

- ウィンドウの現在の幅、高さ
- 凡例、ヘッダー、フッターのサイズ(更に、これらのサイズは、使用されるテキストとフォントに依存します)
- XtNxrtLegendAnchor リソースの値
- ユーザ・プログラムで決められた領域の位置

XRT/graph のデフォルトの位置決めアルゴリズムは、ヘッダー、フッター、凡例のサイズと場所を最初に決定します。グラフは、残りの最も大きい矩形の領域に入るようサイズが変更され、配置されます。

プログラムは位置決めのリソースを使用して、各領域の位置を決定できます。

XtNxrtHeaderX	XtNxrtHeaderY	(XtNxrtHeaderWidth)	(XtNxrtHeaderHeight)
XtNxrtFooterX	XtNxrtFooterY	(XtNxrtFooterWidth)	(XtNxrtFooterHeight)
XtNxrtLegendX	XtNxrtLegendY	(XtNxrtLegendWidth)	(XtNxrtLegendHeight)
XtNxrtGraphX	XtNxrtGraphY	XtNxrtGraphWidth	XtNxrtGraphHeight

(括弧に囲まれたリソースは read-only です)

図 44　位置決めのリソース

*XtGetValues()*コールで使用された場合、位置決めリソースは、グラフが一番最後に表示されたときの値を返します。これらのリソースは、明示的に設定されるか、またはグラフが一度でも表示されるまでは、意味のある値を持ちません。

プログラムは、ウィンドウのサイズが変更されてリソースを再計算する準備ができるまで、位置決めのリソースを設定できません。ウィンドウのリサイズについての詳細は、6.7 をご覧ください。

ある状況においては、位置決めのリソースを一部または全部を設定しておいたほうが良い場合があります。例えば、プログラムがリアルタイムに変化するデータを表示する場合、デフォルトの位置はデータが変化するたびに多少変更されることがあり、障害となることがあります。これには、次のように対処してください。

- 全ての位置決めのリソースをハードコーディングする。この場合ウィンドウのリサイズはできない。または、
- グラフが最初に表示される時、またはウィンドウの大きさが変更された後にグラフが表示されるとき、XRT/graph に全ての領域のデフォルトの位置を計算させる。その後、位置決めリソースを XRT/graph によって計算された値にセットする。

4.6　3D 表示

棒、積み上げ棒、円グラフのデータは、`XtNxrtGraphDepth`、`XtNxrtGraphInclination`、`XtNxrtGraphRotation` を設定することによって、3D 表示にすることができます。[1] これらのリソースはその他のグラフ・タイプには使用できません。

グラフの奥行き(depth)は、グラフの幅に対する割合で設定されます。傾きは X 軸上の視点の位置、回転は Y 軸の右の視点の位置からの角度です。円グラフは回転しません。

[1] `XtNxrtGraphDepth` が 0 より大きければ、回転と傾き(円グラフは傾きのみ)は、エンドユーザがインタラクテイブに調節することができます。詳細は、6.1 を参照してください。

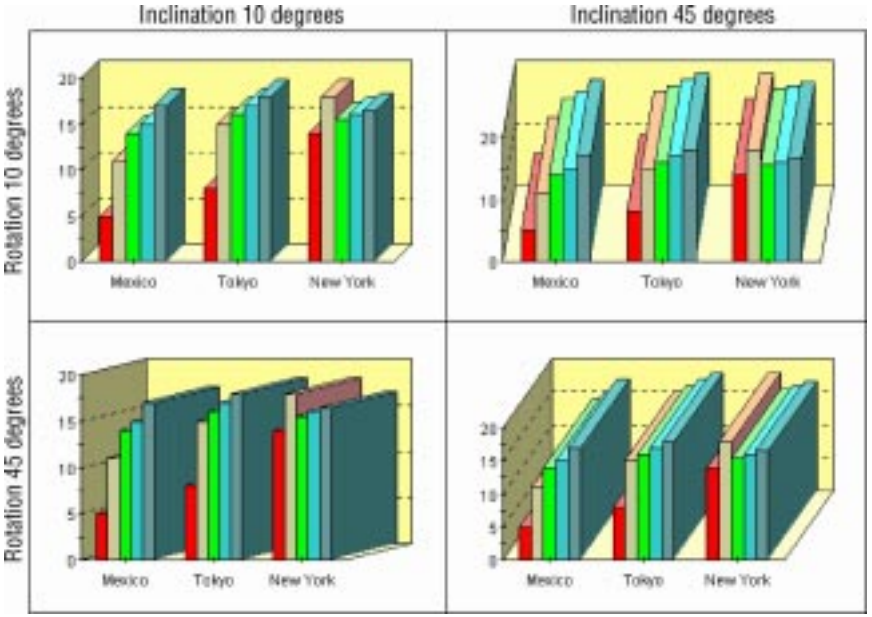


図45　奥行き30の棒グラフの3D表示

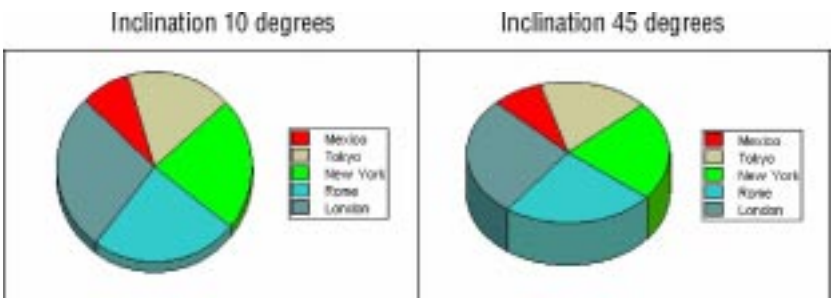


図46　奥行き30の円グラフの3D表示

影の表示

棒と円の 3D 表示には、濃い影と淡い影の 2 種類がデフォルトで割り当てられます。グラフに使用される色数は、`XtNxrtGraph3dShading` リソースによって減らすことができます。`XRT_SHADING_DITHERED` にセットされた場合は、棒や円の色を薄くして、新たな色を使用せずに影の濃淡を表します。`XRT_SHADING_NONE` にセットされた場合は、棒と円の色をそのまま影に使用します。

4.7 スタッキング

XtNxrtIsStacked を True にセットすることにより、線、面、ポーラー、レーダー、filled レーダー等のグラフは、データが配列型の場合にスタッキング・モード(積重ねた形)で表示することができます。XtNxrtIsStacked2 は、複合グラフでの二番目のデータセットに適用されます。除外されたセットまたは series は、スタック(積み上げ)に含まれません。また、非表示になっているセットも除外されている場合と同様に扱われ、スタックには含まれません。尚、負の値の場合は、ポイントが重なり合います。

100%軸

XtNxrt[YY2]Axis100Percent というリソースを使用して、スタッキング・グラフの Y 軸が割合を表すようにすることができます。このリソースは、データが配列型の場合にのみ使用できます。

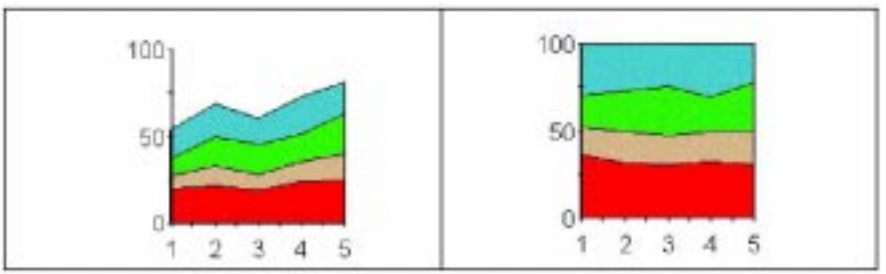


図 47　通常の Y 軸と 100%Y 軸

4.8 ヘッダー、フッター、凡例

特に指定の無い限り、XRT/graph はヘッダーをウィンドウの上部、フッターを下部に配置します。デフォルトの凡例の位置は XtNxrtLegendAnchor に依存します。デフォルトの位置の置き換えに関しては、4.5 を参照してください。

ヘッダーとフッターのテキスト

ヘッダーとフッターは複数行のテキストを表示できます。テキストは XtNxrtHeaderAdjust と XtNxrtFooterAdjust で領域内に行揃えされます。XRT_ADJUST_LEFT、XRT_ADJUST_RIGHT、XRT_ADJUST_CENTER は、それぞれテキストを左揃え、右揃え、中央揃えにします。デフォルトは XRT_ADJUST_CENTER です。

ヘッダーとフッターのテキストは、XtNxrtHeaderStrings と XtNxrtFooterStrings で指定できます。どちらのリソースも NULL 終端文字配列を値として持ちます。以下のコードは 2 行、左揃えのヘッダーを設定します。

```
static char *hs[] = { "Experiment A Results",
     "Last 60 Days", NULL };
XtVaSetValues(graph,
  XtNxrtHeaderStrings,hs,
  XtNxrtHeaderAdjust, XRT_ADJUST_LEFT,
  NULL);
```

グラフのヘッダー、フッター、凡例、グラフ領域の背景にはイメージを表示することができます。また、データやテキスト領域、ヘッダー、フッターは背景をラベルとして使用することができます。但し、グラフ領域、ウィジェットの背景など、テキストを基準とした最小サイズが設定されていない領域や背景では、イメージをラベルとして使用することはできません。

注意: ラベル内のテキストは回転することができますが、イメージは回転できません。

凡例

凡例に表示されるテキストは、セット・ラベルに対応します。(セット・ラベルについては 4.2.2 を参照してください。)

デフォルトでは、XRT/graph は凡例の内容を縦に表示し、グラフ領域の右へ配置します。

凡例のレイアウトは、`XtNxrtLegendOrientation` でコントロールされます。これには、`XRT_ALIGN_VERTICAL` または `XRT_ALIGN_HORIZONTAL` が指定できます。凡例が大きい場合は、数行または数列にレイアウトされます。

グラフ領域に対応するデフォルトの凡例の位置は、`XtNxrtLegendAnchor` でコントロールされます。選択できる値は次のコンパス(方位磁石)の 8 個のポイントに対応します。`XRT_ANCHOR_NORTH`、`XRT_ANCHOR_SOUTH`、`XRT_ANCHOR_EAST`、`XRT_ANCHOR_WEST`、`XRT_ANCHOR_NORTHWEST`、`XRT_ANCHOR_NORTHEAST`、`XRT_ANCHOR_SOUTHEAST`、`XRT_ANCHOR_SOUTHWEST`。

`XtNxrtLegendReversed` は、セットのエントリを逆にします。このリソースを使用して、積上げグラフでは、スタックと同じ上から下の順序に値を凡例に表示させることができます。

`XtNxrtLegendTitle` では、凡例のタイトルを定義することができます。このタイトルは、凡例のフォントを使用し、凡例の上に中央揃えされます。

HiLo、HiLoOpenClose、Candle、Bubble の凡例

凡例のエントリは、セット・ラベルを持つ logical set(論理的なセット)に対して作成されます。logical set に対するラベルは、physical(実際の)インデックスではなく、セットの logical(論理的な)インデックスに基づいています。凡例の各エントリには、logical set に基づいたシンボルと行が線が表示されます。描画されるポイントのサイズは、凡例のフォント・サイズに基づいて決定されますが、ポイント・シンボルの最大サイズより大きくすることはできません。

フォントの指定

ヘッダー、フッター、凡例、軸の注釈には、フォントが指定できます。X サーバでサポートされているフォントであれば、どれでも使用できます。尚、フォントに関する詳細は、7.5 を参照してください。

フォント名を指定する場合は、X Window プログラムの *xlsfonts* または Xlib コールの *XListFonts()* を使用して、サーバでどのフォントが使用可能か確認してください。Xt の手続き、*XtConvertAndStore()*を使用して、フォント・ストリングから XmFontList を取得することができます。フォントの設定に関しては、Xt プログラミングに関する資料を参照してください。[1]

[1] XRT/graph のプロパティ・エディタを使用してフォントを調べたり、選択することができます。

```
static XmFontList
CvtStringToFontList(parent, font_name)
Widget   parent;
char    *font_name;
{
    XrmValue to, from;
    from.addr = font_name;
    from.size = strlen(font_name);
    to.addr = NULL;
    to.size = 0;
    if (!XtConvertAndStore(parent, XmRString, &from,
             XmRFontList, &to)) {
        fprintf(stderr, "Conversion of %s to XmFontList \
                failed!\n", font_name);
        return((XmFontList)0);
    }
    return(*(XmFontList *)to.addr);
}

XmFontList hf, ff;
hf=CvtStringToFontList(graph, "-*-times-medium-*-*--18-*-iso8859-*");
ff=CvtStringToFontList(graph, "9x15");
if (hf != NULL && ff != NULL) {
    XtVaSetValues(graph,
        XtNxrtHeaderFont, hf,
        XtNxrtFooterFont, ff,
        NULL);
}
```

図48　XmFontList リストの設定

4.9　データ・スタイル

データがどのように表示されるか(色、線/塗りつぶしのパターン、ポイント・スタイル、線幅等)は、データ値に定義されたデータ・スタイルに依存します。例えば、三番目のデータ・セットの値は、三番目のデータ・スタイルを使用して表示されます。

デフォルトのデータ・スタイル

XRT/graph は、モノクロとカラーのサーバ用のデータ・スタイルを提供しています。XtNxrtDataStylesUseDefault が True であれば、グラフには最適なデフォルトのデータ・スタイルが使用されます。データ・スタイルを詳細に設定した後でも、XtNxrtDataStylesUseDefault を True にセットすれば、デフォルトのデータ・スタイルを使用することができます。

XrtDataStyle 構造体は、*<Xm/xrtgfcom.h>*に定義されています。

```
typedef struct {
  XrtLinePattern   lpat;    /* line pattern */
  XrtFillPattern   fpat;    /* fill pattern */
  char *       color;   /* color */
  int          width;   /* line width */
  XrtPoint     point;   /* point style */
  char *       pcolor;  /* point color */
  int          psize;   /* point size */
  unsigned long    res1;    /* reserved */
  unsigned long    res2;    /* reserved */
  unsigned long    resh;    /* reserved */
  unsigned long    ress;    /* reserved */
} XrtDataStyle;
```

XrtDataStyle データ構造体は、データの表示に関する情報を持っています。フィールドは次の通りです。

lpat	*<Xm/xrtgfcom.h>*に定義される、線のパターン。図 49 にある XRT_LPAT_constants から選択する。
fpat	*<Xm/xrtgfcom.h>*に定義される、面、棒、円グラフの塗りつぶしパターン。図 49 にある XRT_FPAT_constants から選択する。
color	線や、面、棒、円グラフの塗りつぶしに使用される色。色名か、'#'で始まる RGB(赤、緑、青)構成の二桁の 16 進数の文字列で指定する。色の指定の詳細については、2.8 を参照。
width	線グラフに使用される線の太さ。1 以上の値を指定する。
point	*<Xm/xrtgfcom.h>*に定義される、ポイント・スタイル。図 51 にある XRT_POINT_constants から選択する。
pcolor	ポイントに使用される色。色名か、'#'で始まる RGB(赤、緑、青)構成の二桁の 16 進数の文字列で指定する。色の指定の詳細については、2.8 を参照。
psize	ポイントのサイズ。0 以上の値を指定する。0 の場合は、ポイントが表示されない。サイズは相対的に測られるので、サイズ 12 が 12 pixel の大きさになるとは限らない。

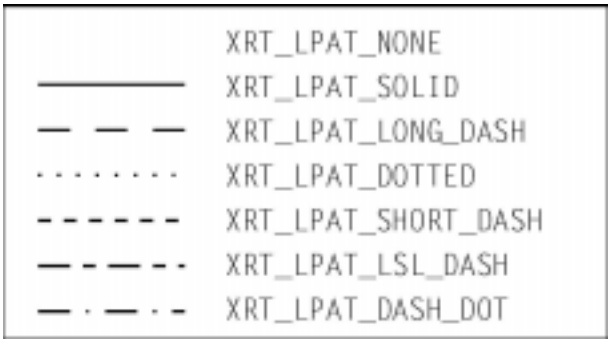


図49　線パターン

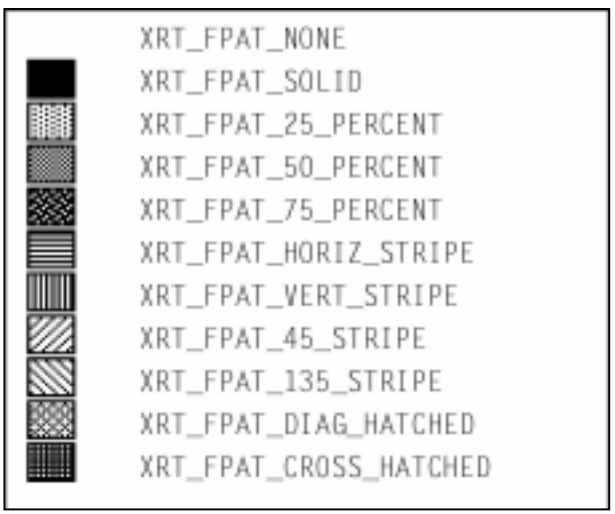


図50　塗りつぶしパターン

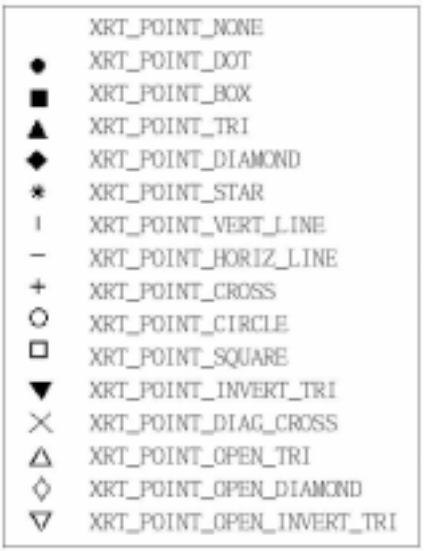


図51　ポイント・スタイル

データスタイルの設定

XRT/graph は、デフォルトで最適なデータ・スタイルを選択します。データ・スタイルを詳細に設定する場合は、そのスタイルが使用するディスプレイに適するか注意してください。例えば、濃い赤の棒と濃い緑の棒はモノクロのモニタでは同じに見えてしまいます。

データ・スタイルを設定すると、`XtNxrtDataStylesUseDefault` が `False` になります。XRT/graph は、何も設定されていないスタイルには、デフォルトのスタイルを使用します。図52は、プログラムでデータ・スタイルをセットする例を示しています。

```
static XrtDataStyle color_styles[] = {
/* 0 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, "red", 1, XRT_POINT_TRI,
    "magenta", XRT_DEFAULT_POINT_SIZE },
/* 1 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, "tan", 1, XRT_POINT_BOX,
    "green", XRT_DEFAULT_POINT_SIZE },
/* 2 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, "LimeGreen", 1, XRT_POINT_DOT,
    "medium blue", XRT_DEFAULT_POINT_SIZE },
/* 3 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, "MediumTurquoise", 1,
    XRT_POINT_DIAMOND, "red", XRT_DEFAULT_POINT_SIZE },
/* 4 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, "blue", 1, XRT_POINT_STAR,
    "red", XRT_DEFAULT_POINT_SIZE },
/* 5 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, "magenta", 1, XRT_POINT_TRI,
    "medium blue", XRT_DEFAULT_POINT_SIZE },
/* 6 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, "green yellow", 1, XRT_POINT_BOX,
    "magenta", XRT_DEFAULT_POINT_SIZE },
/* 7 */ { XRT_LPAT_SOLID, XRT_FPAT_SOLID, "medium blue", 1, XRT_POINT_DOT,
    "light sky blue", XRT_DEFAULT_POINT_SIZE },
};

static XrtDataStyle *my_dstyles_ptr[] = {
    &color_styles[0],
    &color_styles[1],
    &color_styles[2],
    &color_styles[3],
    &color_styles[4],
    &color_styles[5],
    &color_styles[6],
    &color_styles[7],
    NULL
};
...
    Widget graph;
...
    graph = XtVaCreateManagedWidget("my_graph",
        xtXrtGraphWidgetClass, parent,
...
    XtNxrtDataStyles, my_dstyles_ptr,
    NULL );
```

図52　データ・スタイルの設定

データ・スタイルへのアクセス

XtNxrtDataStyles リソースまたは、*XrtSetNthDataStyle()*、*XrtGetNthDataStyle()*を使用して、データ・スタイルへアクセスできます。

XtNxrtDataStyles は、データ・スタイルの配列全体を取得したり、設定したりする場合に使用できます。*XtGetValues()*に使用された場合は、返された配列ポインタは read-only です。*XrtDupDataStyles()*簡易ルーチンを使用することにより、返されたデータ・スタイルの配列ポインタを複写することができます。

例えば、全ての線幅を 5 pixel にセットする場合、次のコードが使用できます。

```
int i;
XrtDataStyle **ds, **myds;
XtVaGetValues(graph, XtNxrtDataStyles, &ds, NULL);
myds = XrtDupDataStyles(ds);
for(i=0; myds[i]; i++){
  myds[i]->width = 5;
}
XtVaSetValues(graph,
  XtNxrtDataStyles, myds,
  NULL);
XrtFreeDataStyles(myds);
```

*XrtGetNthDataStyle()*を使用して、特定な `XrtDataStyle` 構造体のポインタを取ることができます。例えば、`graph` が図 30 にある人口データのグラフなら、以下のコードは、Tokyo のデータの線の太さをプリントし、それを 2 倍にします。

```
XrtDataStyle *dstyle, my_dstyle;
dstyle = XrtGetNthDataStyle(graph, 1);
printf("line width for 2nd set of data: %d\n", dstyle->width);
my_dstyle = * dstyle;
my_dstyle.width = my_dstyle.width * 2;
XrtSetNthDataStyle(graph, 1, &my_dstyle);
```

軸のデータ・スタイル

`XtNxrt[XYY2]AxisDataStyle` と `XtNxrt[XYY2]AxisDataStyleUseDefault` を使用して、各軸は、それぞれ異なる色、線幅やパターンで表示できます。

2 本の Y 軸を持つグラフで、Y2 軸が一番目の軸に従属する時、Y2 軸は Y 軸のデータ・スタイルを使用し、Y1 軸が Y2 軸に従属する場合は、Y1 軸が Y2 軸のデータ・スタイルを使用します。

PointStyle のデータ・スタイル

PointStyle は、特定なデータ・ポイントを変更する時に使用できます。PointStyle を使用すると、他とは異なるデータ・ポイントを設定したり、円グラフのスライスを円から切り離して表示することができます。PointStyle オブジェクトは、`XrtPointStyleHandle` 型のハンドルによって参照されます。下記は、特定なポイントに対してスタイルを割り当てるルーチンの一覧です。Appendix C にも、これらの解説がありますので参照してください。

PointStyle オブジェクトのルーチン	
XrtPointStyleCreate()	PointStyle オブジェクトをグラフの子として作成する
XrtPointStyleDestroy()	PointStyle オブジェクトを破棄する
XrtPointStyleSetValues()	PointStyle オブジェクトの属性の現在値をセットする
XrtPointStyleGetValues()	PointStyle オブジェクトの属性の現在地を取得する

注意: PointStyle は、使用する前に必ず、XRT_POINTSTYLE_SET と XRTPOINTSTYLE_POINT をセットしてください。

インタラクテイブなデータ・スタイルの編集

XRT/graph のプロパティ・エディタを使用して、インタラクティブにデータ・スタイルを編集することができます。*feedback* と *vmgraph* デモには、データ・スタイル・エディタが含まれています。このソースコードは、*$XRTHOME/src/graph/tools/mtgraph.c* にあります。

4.10 棒グラフの特別なリソース

棒グラフは、棒の表示や間隔をコントロールするリソースを持っています。グラフの棒はクラスタ (*clusters*)にグループ化され、各ポイントにそれぞれクラスタがあります。

積上げ棒グラフ

XtNxrtType が XRT_TYPE_STACKING_BAR にセットされている時、データは積上げ棒グラフで表示されます。一つのクラスタが、一本の棒を持ちます。0 より小さい Y 値は、X 軸の下方向に表示されます。

クラスタの幅

棒グラフでは、棒はクラスタにグループ化されています。棒がクラスタのスペースをどれだけ占めるかを指定するには、XtNxrtBarClusterWidth を使用します。デフォルトでは XtNxrtBarClusterWidth は 50 で、棒がクラスタの幅の 50%のスペースを占めています。

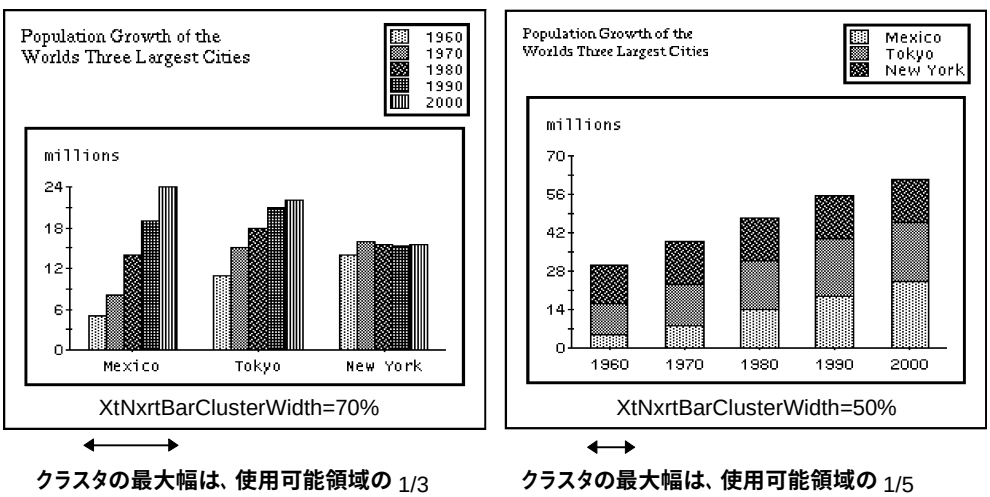


図 53 棒グラフと積上げ棒グラフのクラスタ幅

棒の重なり

同じクラスタ内での各棒の間隔は、XtNxrtBarClusterOverlap でコントロールされます。クラスタでの重なり合いは棒の幅に対する割合で指定します。積み上げ棒グラフでは、一つのクラスタにつき一本の棒が表示されるので、このリソースは意味を持ちません。

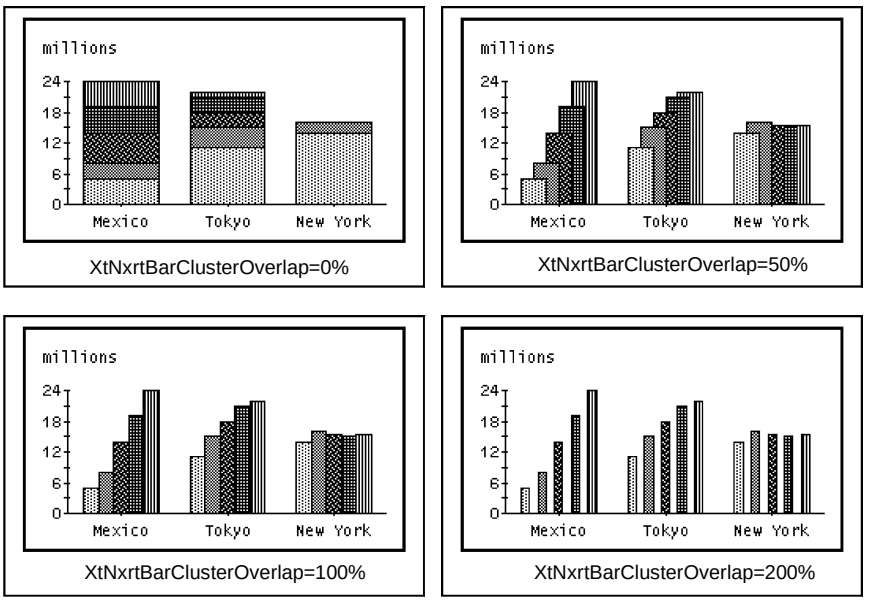


図54　クラスタの重なり

積上げ棒のパーセンテージ表示

積上げ棒グラフの Y 軸は、XtNxrt[YY2]Axis100Percent リソースを使用して、各データの割合を示すパーセンテージ表示に切り替えることができます。このリソースが True にセットされると、各積上げ棒の Y 値の合計が 100%とみなされ、棒の各部分の Y 値の全体に対する割合が計算されます。このリソースは、棒グラフには使用できません。

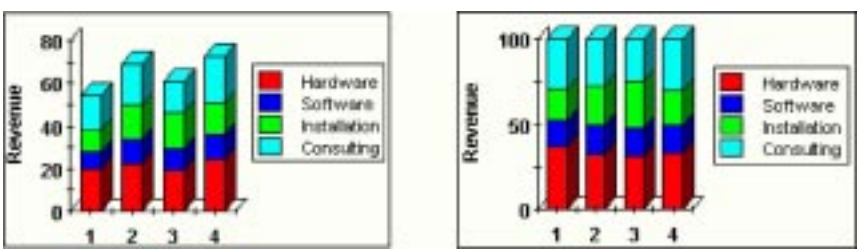


図55　積上げ棒 - 普通の場合と、100%Y 軸の場合

4.11 円グラフの特別なリソース

円グラフは、線や棒グラフにある 2 次元の座標がないという点で、他のグラフと大きく異なります。また、円グラフには、指定されたしきい値より小さい値をグループ化する*"その他"*(*other*)という分類があります。

XRT/graph は、全てのセットの各ポイントを比較しながら円グラフを作成します。はじめに、全ての一番目のポイント(次に二番目、三番目..と続きます)の合計が計算されます。そして、各ポイントの合計に対する割合が計算されます。多数のポイントを持つデータ・オブジェクトは円グラフには適しません。また、負の値は円グラフでは、0 として扱われます。

あるセットの中で各ポイントのサイズを比較するには、*XrtDataTranspose()*を使用してデータ・セットをトランスポーズしてください。(詳細は 5.5 に解説されています。)円グラフのこれらの特性のため、XRT/graph は、`XtNxrtType` が `XRT_TYPE_PIE` の時のグラフにのみ適用されるリソースを定義します。

"その他"のスライス

しきい値には、*しきい値法*(*threshold method*)と*しきい値*(*threshold value*）という二つの要素があります。これらは、どの値を個別のスライスで表示し、どの値が"その他"のスライスにグループ化されるかを決定します。

`XtNxrtPieThresholdMethod` が `XRT_PIE_SLICE_CUTOFF` であれば、XRT/graph は `XtNxrtPieThresholdValue` の値を調べ、合計に対する割合がしきい値より小さいデータは、"その他"のスライスへグループ化されます。`XtNxrtPieThresholdMethod` が `XRT_PIE_PERCENTILE` であれば、最小値から幾つかの値を"その他"のスライスへグループ化し、そのスライスがしきい値と同じか小さくなるようにします。

`XtNxrtPieThresholdValue` には、0.0 から 100.0 の間の浮動小数点値が定義できます。"その他"のスライスが作成されないようにするには、`XtNxrtPieThresholdValue` を 0.0 にセットしてください。

*XrtGetOtherSliceSets()*を使用して、どのセットが"その他"のスライスへグループ化されているか調べることができます。この方法は、Appendix D に解説されています。

"その他"のスライスのラベル

"その他"のスライスのラベルの指定には、`XtNxrtOtherLabel` を使用します。デフォルトでは、"その他"のスライスには、"Other"というラベルが付きます。このリソースを使用して、ロケールに適したラベルを指定することができます。

"その他"のスライスのデータ・スタイル

`XtNxrtOtherDataStyle` は、"その他"のスライスのデータ・スタイルを指定します。`XrtDataStyle` については、4.9 を参照してください。デフォルトでは、XRT/graph は"その他"のスライスには、他と異なったスタイルを選択します。`XtNxrtOtherDataStyleUseDefault` を `True` にすると、アプリケーションは、"その他"のスライスにデフォルトのデータ・スタイルを使用することができます。

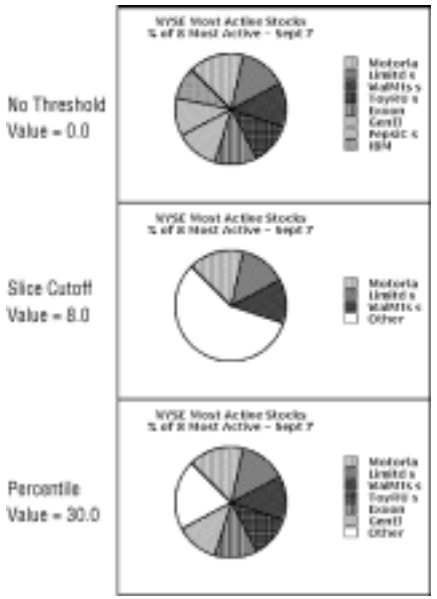


図56　しきい値法

スライスの順序

スライスが円に配置される順序は、XtNxrtPieOrder によって決定されます。XRT_PIEORDER_ASCENDING 、 XRT_PIEORDER_DESCENDING そして、XRT_PIEORDER_DATA_ORDER の中から選択してください。XRT_PIEORDER_DATA_ORDER が選択された場合、"その他"のスライス以外はデータの順に表示されます。"その他"のスライスは常に最後に配置されます。

スライスの最小数

XtNxrtPieMinSlices は、表示されるスライスの最少数を指定します。このリソースはしきい値法より優先されます。もし、XtNxrtPieMinSlices が 5 に設定してあれば、XRT/graph は、データを"その他"にグループ化する前に 5 個のスライスを表示しようとします。

全てが 0 の円

もし、円の全てのデータ値が 0(または負の値)の場合、円は作成されずマーカも設定できません。*XrtPick()*は、(円が作成されていると仮定して)正しい円のインデックスを示しますが、スライスのインデックスは常に 0 となります。*XrtPick()*は、全てが 0 の円には非 0 の間隔値を返します。

スライスが切り離された円(exploding pie)

円グラフを構成するスライスは、PointStyle のメンバ XRT_POINTSTYLE_SLICE_OFFSET を使用して円から離して表示させることができます。円からのオフセットは、円の半径に対する割合で表されます。

このオフセットや PointStyle に関連するリソースについては、P79 の PointoStyle のデータ・スタイルの解説を参照してください。

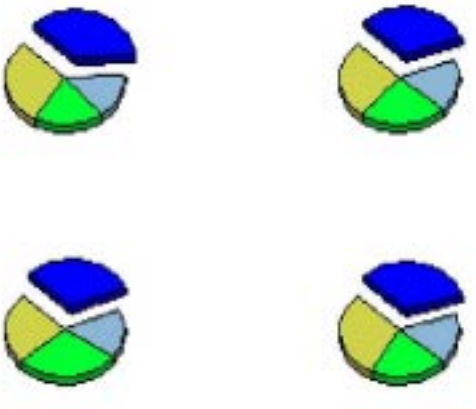


図57　スライスが切り離された円

円の開始点

デフォルトでは、全ての円は、135.0 度を開始点として描画します。この値は、XtNxrtPieStartAngle によって変更することができます。

4.12　HiLo とキャンドル・グラフとバブル・チャート

HiLo、HiLoOpenClose、キャンドル、バブル等のグラフは、各棒やバブルを構成するために複数のセットの Y 値を使用します。HiLo とバブルは 2 セットづつ、HiLoOpenClose とキャンドルは 4 セットづつ使用します。

これらのセットは、*論理的なセット(logical set)*を構成し、セット・ラベルやマーカ、テキスト領域等はこのセットに指定します。グラフ・タイプが HiLo、HiLoOpenClose、キャンドル、バブルの時は、ほとんど全てのリソース、手続きや方法が、この *logical set* を使用します。但し、データ・オブジェクトとデータ・スタイルは例外です。

バブル・チャートのリソース

バブル・チャートでは、XtNxrtBubbleMethod がバブル値の表示に使用されます。このリソースの値は下記の表の通りです。

XRT_BUBBLE_METHOD_DIAMETER　バブルの直径を変更することによりデータを表示する。デフォルト。

XRT_BUBBLE_METHOD_AREA　　バブルの領域を変更することによりデータを表示する。

バブル・チャートでは、各ポイントに対して三つの値が必要です。logical set(論理的なセット)の各ポイントに対する X と Y 値は、一番目の physical set(実際のセット)で対応するポイントの X と Y 値から取得されます。各ポイントの三番目の値は、二番目の physical set で対応するポイントの Y 値から取得されます。二番目の physical set の X 値は無視されます。このようにして構成されたポイントは、*バブル値(bubble value)*として参照されます。

XtNxrtBubbleMin と XtNxrtBubbleMax は、ポイント・シンボルの直径/領域の最小値と最大値をコントロールします。これらのリソースは、グラフ領域全体に対する直径/領域のパーセンテージとして指定される XrtFloat 値です。グラフ領域の直径は、グラフ領域の高さまたは幅の内小さい方で定義され、グラフ領域は高さに幅を掛けた値として定義されます。これらの二つのリソースは、バブル値の最小/最大値と一緒に使用され、各ポイントに対して描画されるシンボルの直径/領域を決定します。XtNxrtBubbleMin の値が XtNxrtBubbleMax の値より大きい場合は、バブル値が増加するにつれてポイント・シンボルの直径/領域が減少していきます。ポイント・シンボルの直径/領域は、バブル値の範囲に依存するだけですので、バブル値は必ずしも全てが正の値または負の値である必要はありません。

バブル値の間には、線グラフと同様に線を描画することができます。各 logical set に対する線のパターンや色、幅は、一番目の physical set のデータ・スタイルから取得されます。ポイント・スタイルは、バブル値の基本的な形を決定します。デフォルトは triangle(三角)ですが、円を使用したバブル・チャートの方が一般的です。円を使用するには、ポイント・スタイルを XRT_POINT_DOT にしてください。ポイント・シンボルのその他の要素は、データ・スタイルであるポイントの色、背景色、塗りつぶしパターン及び、XtNxrtGraphShowOutlines、XtNxrtGraphForegroundColor とバブル値が決定します。これらのうち、どれが実際に使用されるかは、多くの場合、ポイント・スタイルに依存します。

シンボルの描画法

XRT_POINT_DOT XRT_POINT_BOX XRT_POINT_TRI XRT_POINT_DIAMOND XRT_POINT_INVERT_TRI	■ **描画されるシンボルには、背景色、ポイント色、塗りつぶしパターンが使用される。**

XRT_POINT_STAR XRT_POINT_CROSS XRT_POINT_SQUARE XRT_POINT_CIRCLE XRT_POINT_VERT_LINE XRT_POINT_HORIZ_LINE XRT_POINT_DIAG_CROSS XRT_POINT_OPEN_TRI XRT_POINT_OPEN_DIAMOND XRT_POINT_OPEN)INVERT_TRI	**シンボルを構成する線は、ポイント色を使用する。** **線幅は、シンボルの直径を 7 で割った値に等しい。**

直径と領域の定義

XRT_POINT_DOT XRT_POINT_BOX XRT_POINT_TRI XRT_POINT_DIAMOND XRT_POINT_INVERT_TRI XRT_POINT_SQUARE XRT_POINT_CIRCLE XRT_POINT_OPEN_TRI XRT_POINT_OPEN_DIAMOND XRT_POINT_OPEN_INVERT_TRI	**直径は、シンボルの高さとして定義される。** **領域は、実際のシンボルが占める領域として定義される。**
XRT_POINT_STAR XRT_POINT_CROSS XRT_POINT_VERT_LINE XRT_POINT_HORIZ_LIINE XRT_POINT_DIAG_CROSS	**直径は、線の長さとして定義される。** **領域は、上記の直径を使用した円の占める領域として定義される。**

HiLoOpenClose リソース

グラフ・タイプが XRT_TYPE_HILO_OPEN_CLOSE の時、次のリソースが open と close の tick マークをコントロールします。

XtNxrtHiLoOpenShow	open の tick マークを表示、非表示にする。
XtNxrtHiLoCloseShow	close の tick マークを表示、非表示にする。
XtNxrtHiLoOpenCloseFullWidth	棒の幅全体に渡って open と close の tick マークを表示する。

XtNxrtCandleComplex

キャンドル・グラフには、XtNxrtCandleComplex で指定される二つのスタイルがサポートされています。このリソースが False にセットされていると、最初のセットのデータ・スタイルがキャンドルの表示をコントロールします。透明なキャンドルは、株価の下降を示し、塗りつぶされたキャンドルは株価の上昇を示します。スタイルには、色、幅、線パターン、塗りつぶしパターン、ポイント・サイズが使用されます。

このリソースが True にセットされると、四つの全てのセットのデータ・スタイルがキャンドルの表示をコントロールします。

■ 最初のデータ・スタイルの色、幅、線パターンが HiLo ラインに使用される。

- 二番目のデータ・スタイルの色、幅、塗りつぶしパターンが上昇株価のキャンドルに使用される。
- 三番目のデータ・スタイルの色、幅、塗りつぶしパターンが下降株価のキャンドルに使用される。
- 四番目のデータ・スタイルの色が、キャンドルのアウトラインに使用される。

尚、データ・スタイルのカスタマイズについては、4.9 を参照してください。



図 58　キャンドル・グラフ

4.13　ポーラー/レーダー・グラフの特別なリソース

ポーラー、レーダー、filled レーダーは、X 原点や Y の注釈をコントロールする幾つかのリソースを持っています。これらのグラフ・タイプでは、X 軸はサークルで、Y 軸はサークルの中心からの距離を測定します。

ユニット

`XtNxrtAngleUnit` は、X 値と X 軸のリソース値を解釈する時に使用されるユニットを指定します。デフォルトでは、degree が使用されます。X 値を radian で解釈する時は `XRT_ANGLE_RADIANS` に、grads の時は `XRT_ANGLE_GRADS` にセットします。

X 軸の“原点”

X 軸の”原点”はサークルの 0 度の位置にあり、ポーラー・グラフでは(デフォルトで)サークルの 3 時の位置、レーダーと filled レーダーでは 12 時の位置にあります。`XtNxrtXOriginBase` を使用して、0 度の位置を指定することができます。指定された値は、`XtNxrtAngleUnit` のユニットで解釈されます。

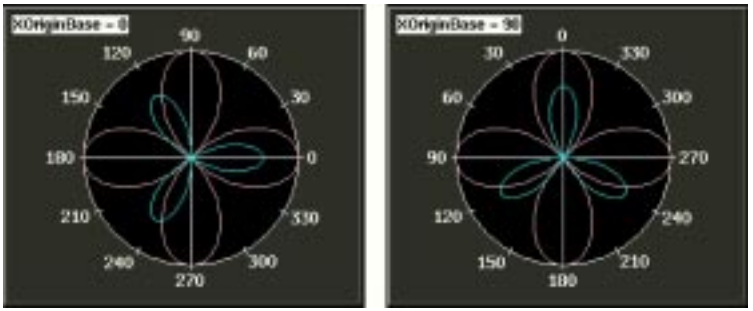


図59　X 軸の原点ベースが 0 と 90 の場合

注釈の角度

Y 軸の注釈は、デフォルトで X 原点の Y 軸に沿って表示されます。注釈は、XtNxrt[YY2]AnnotationAngle を使用して、サークルの任意のポイントに配置できます。

ポーラー・グラフの X 軸の範囲

XtNxrtPolarHalfRange という Boolean リソースを使用して、全範囲のポーラー・グラフ(0 から 360 度)と 1/2 範囲のグラフ(-180 から 180 度)に切り替えることができます。

ポーラー・グラフの負の値

XtNxrtPolarAxisAllowNegatives は、ポーラー・グラフが負の値を持つことができるかどうかを指定します。

4.14　前景色と背景色

背景の色

ウィンドウの背景の色は、XtNxrtBackgroundColor によって指定されます。デフォルトは白です。

ヘッダー、フッター、凡例、グラフの各領域は、それぞれ背景色を持ち、デフォルトは NULL(無色)です。例えば、凡例の背景色は、XtNxrtLegendBackgroundColor で変更することができます。グラフのデータ領域も独自の背景色を持つことができ、XtNxrtDataAreaBackgroundColor で指定します。

モノクロのサーバでは、黒以外の背景色は白で表示されます。

前景の色

ウィンドウの前景の色はデフォルトで黒ですが、XtNxrtForegroundColor で変更できます。ヘッダー、フッター、凡例、グラフの各領域の前景にも、他の色が別に指定されていない限り、このリソースの色が使用されます。他の色を指定する場合は、例えばヘッダーでは XtNxrtHeaderForegroundColor を使用します。

グラフのデータ領域も独自の背景色を持つことができ、XtNxrtDataAreaForegroundColor[1]で指定します。

モノクロのサーバでは、白以外の前景色は黒で表示されます。

背景/前景色の指定

全ての前景と背景の色のリソースは、色のストリング(2.8 参照)を値として取ります。パステルカラーの"LightBlue"や"Wheat"等は背景色に、濃い色は前景色に適しています。以下はその例です。

```
XtVaSetValues(my_graph,
  XtNxrtBackgroundColor, "LightBlue",
  XtNxrtForegroundColor, "NavyBlue",
  XtNxrtHeaderBackgroundColor, "LightGray",
  XtNxrtFooterForegroundColor, "DarkGreen",
  NULL);
```

グラフのデータ表示に使用されている色を背景に選択しないように注意してください。次に挙げる色は、デフォルトのデータ・スタイル色と重なるおそれがありますので、デフォルトが使用されている時は、選択しないでください。aquamarine, blue, cadet blue, gold, green, green yellow, khaki, light blue, light sky blue, lime green, magenta, medium blue, medium orchid, medium turquoise, medium violet red, plum, olive drab, orange, red, sky blue, steel blue, tan, yellow green

4.15 マーカ

マーカを使用することにより、スクリーン上の特定なデータ要素や、グラフの特定なデータ値を識別できるようになります。マーカには多数の利用法がありますが、最もよく使用されるのは、ユーザがマウスでグラフのある箇所を選択した時に、そのポイントに関するフィードバックを提供する機能です。

線、面、棒、積上げ棒、HiLo、HiLoOpenClose、キャンドル・グラフは、X と Y マーカをサポートしています。積上げ棒グラフ以外では、X マーカはグラフ領域に引かれる X 軸に垂直な線です。積上げ棒グラフでは、X マーカが XRT_MARKER_SET_POINT であれば、棒のブロックに描かれる短い線、それ以外の場合は X 軸に垂直な線です。Y マーカは、グラフ領域に引かれる Y 軸に垂直な線です。

円、ポーラー、レーダー、filled レーダーは、一つのマーカをサポートしており、これは、グラフの中央から選択されたデータ・ポイントまたはスライスの中心を通る線です。

複合グラフ(7.2 参照)では、マーカが対応するデータ・セットを XtNxrtMarkerDataset で指定してください。

[1] このリソースは、XtNxrtGraphShowOutlines が False の時は無視されます。

マーカのデータ・スタイル

デフォルトのマーカは、黒、1 ピクセル幅の破線です。XtNxrtMarkerDataStyle を使用して、X、Y 両方のマーカのスタイルを指定できます。

XtNxrt[XY]MarkerDataStyle を使用すると、各マーカのスタイルを別々に指定できます。マーカは、XrtDataStyle 構造体の線パターン、色、線幅を使用します。(データ・スタイルの設定法は、4.9 に解説されています。

XtNxrt[XY]MarkerDataStyle リソースが明示的に設定されていなければ、XtNxrtMarkerDataStyle が使用されます。XtNxrt[XY]MarkerDataStyle が設定されていれば、その値が使用されますが、対応する UseDefault を True にセットすれば、デフォルトの値に戻ります。

マーカのパフォーマンス

マーカの位置決めのリソースのアップデートは非常に速く行われます。これは、XRT/graph がマーカの位置を変える時に、ウィンドウ全体を再描画しないからです。マーカのパフォーマンスは、XtNxrtDoubleBuffer を True にセットすると向上します。但し、グラフのパフォーマンスはダブル・バッファリングをオフにした方が良い結果が得られます。

CPU、ネットワーク、X サーバの性能によっては、マーカでマウスの動きをリアルタイムで追うことができます。マーカによるマウスの追跡については、*feedback* と *scroll_time* デモ・プログラムに例がありますので、参照してください。

X マーカの位置

XtNxrtXMarkerMethod は、X マーカの位置決めの方法をコントロールします。このリソースは、次の値にセットできます。

XRT_MARKER_SET_POINT	XtNxrtXMarkerSet と XtNxrtXMarkerPoint が X マーカの位置を指定。円、レーダ、filled レーダーには、この方法しか使用できない。
XRT_MARKER_VALUE	XtNxrtXMarker が X マーカの位置を指定する。

XtNxrtXMarkerSet と XtNxrtXMarkerPoint は、データ・オブジェクトのインデックスで、X マーカが描画されるデータ・ポイントを定義します。

XtNxrtXMarker 浮動小数点リソースの値は、マーカが描画される X 軸の値です。

Y マーカの位置

Y マーカは、XtNxrtYMarker 浮動小数点リソースで位置決めされます。これは、Y マーカが描画される Y 軸の値を指定します。このリソースは円、ポーラー、レーダー、filled レーダーのグラフでは無視されます。

X、Y マーカは、XtNxrt[XY]MarkerShow が True にセットされていなければ、表示されません。

4.16 枠線

ヘッダー、フッター、凡例、グラフの各領域は枠線で囲むことができます。各領域の枠線は Border と BorderWidth のリソースで指定できます。(例: XtNxrtGraphBorder と XtNxrtGraphBorderWidth) また、グラフ全体を囲む枠線も指定することができ、この線幅は、XmNshadowThickness でコントロールできます。

Border リソースは、XRT_BORDER_NONE、XRT_BORDER_PLAIN、XRT_BORDER_3D_OUT、XRT_BORDER_3D_IN 、 XRT_BORDER_ETCHED_IN 、 XRT_BORDER_ETCHED_OUT 、 XRT_BORDER_FRAME_IN 、 XRT_BORDER_FRAME_OUT 、 XRT_BORDER_BEVEL 、 XRT_BORDER_SHADOW のどれかに設定できます。線幅は 0 から 20 の間です。

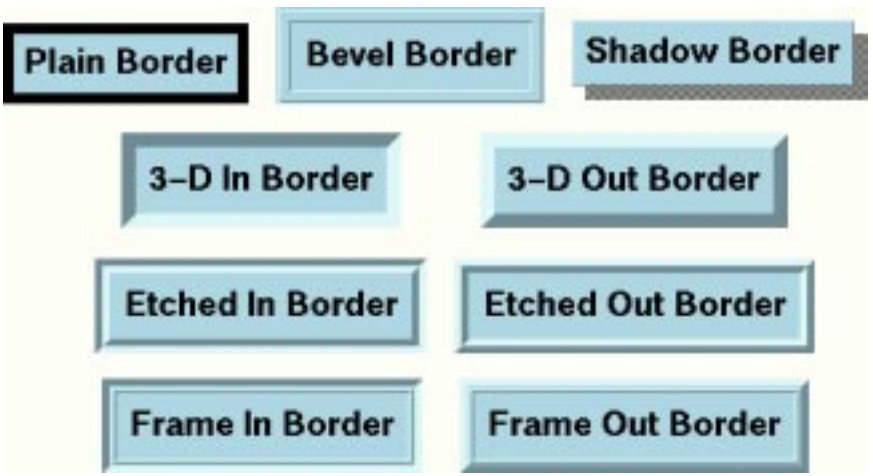


図 60　枠線の種類

4.17 背景のイメージ

ヘッダー、フッター、凡例、グラフ、データ領域の背景にはイメージを表示することができます。これらの各領域に対して、(XtNxrtGraphImage、XtNxrtGraphImageLayout のように)Image と Image Layout というリソースが用意されています。イメージは、ウィジェット全体に対して表示させることもできます。

Image リソースは XrtImage 型で、この型は XrtPDSImage 型(8.1 参照)と同じです。

次の例では、XRT/graph の変換プログラムを使用してフッターの背景にイメージを表示しています。

```
#include <Xm/XrtPDSImageObj.h>

void
set_image(Widget graph, char *file_name)
{
      XrtImage image;
      XrmValue from, to;

      from.addr = file_name;
      from.size = strlen(file_name) + 1;
      to.addr = (caddr_t)&image;
      to.size = sizeof(image);

if (!XtConvertAndStore(graph, XtRString, &from, XtRXrtImage, &to)) {
      fprintf(stderr, "Conversion of %s to XtRXrtImage failed\n",
               file_name);
      image = NULL;
}
XtVaSetValues(graph,
               XtNxrtFooterImage, image,
      NULL);
 XrtPDSImageDestroy((XrtPDSImage)image);
}
```

イメージはウィジェットによってコピーされますので、変換プログラムによって作成されたイメージは、必ず破棄してください。イメージを取り去るには、対応するイメージのリソースを NULL にセットしてください。

イメージは、*XrtSetPropString()*を使用してプロパティを設定することにより、セットすることもできます。

```
XrtSetPropString(graph, XRT_FOOTER_IMAGE,
      "$XRTHOME/src/icons/shading.xpm");
```

この機能は、*XtConvertAndStore()*ではなく、XRT/graph の変換プログラムを直接に使用し、ウィジェットにイメージをセットする処理は、全て内部で行われます。また、(イメージがリソース・ファイルで指定されように)$XRTHOME 環境変数が解釈されます。

サポートされているイメージのフォーマットや、任意のフォーマットを使用する方法に関しては、8.1 を参照してください。

Image Layout リソースは、背景のイメージのサイズや位置を指定します。次の値は、設定できる配置です。

- XRT_IMAGELAYOUT_CENTERED は、領域の中心にイメージをそのまま配置
- XRT_IMAGELAYOUT_TILED は、イメージをタイルのように配置して領域全体を占めるようにする

領域に指定されるイメージは、そこに指定されている背景色より優先されます。但し、ヘッダーに背景色が指定されている場合は、色がイメージを上書きします。

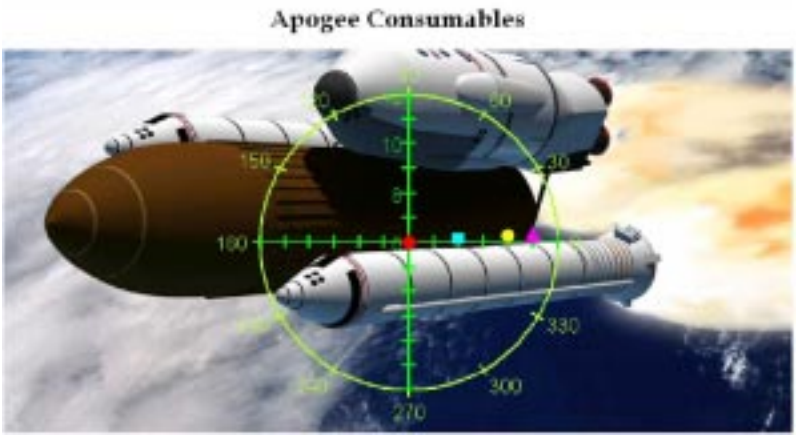


図61　グラフの背景に表示されたイメージ

イメージの最小サイズ

背景のイメージは、テキスト領域やヘッダー、フッターにテキストの代わりに使用することができます。`XtNxrtTextImageMinimumSize`、 `XtNxrtHeaderImageMinimumSize` 、`XtNxrtFooterImageMinimumSize` のリソースが `True` にセットされると、イメージと同じサイズのラベルが作成されます。

この機能を使用して、例えば、`XtNxrtTextImageMinimumSize` をテキスト領域に使用し、カスタマイズしたポイントをグラフの任意の場所に配置することができます。

4.18　ダブル･バッファリング

ダブル･バッファリングは、グラフが変化する時のちらつきを減少させます。

`XtNxrtDoubleBuffer` が `True` の場合、グラフが変化する時に XRT/graph は次のような働きをします。

- サーバに Pixmap を割り当てる(または消去する)
- 完成されたグラフを Pixmap に書く
- サーバに Pixmap を画面にコピーするよう命令する

`XtNxrtDoubleBuffer` が `False` の時は、XRT/graph はグラフが変化する度に、画面のイメージを消去し(ちらつきが起こります)、そして完成したイメージをウィンドウに表示します(グラフが一部分づつ表示されるように見えます)。

デフォルトでは XtNxrtDoubleBuffer は True ですが、False に設定するとグラフのパフォーマンスが向上し、サーバに要求するメモリが減少します。

4.19 スクロールバー

グラフ・ウィジェットは、ScrolledWindow ウィジェットに配置することができます。ユーザは、スクロールバーを使用してウィンドウで表示しきれない部分のグラフを見ることができます。ScrolledWindow は、スクロール機能をコントロールし、グラフを自動的に再配置します。

以下のコードは、その方法を示しています。始めに、XmNscrollingPolicy が XmAUTOMATIC にセットされて、ScrolledWindowWidget が作成されます。次にグラフが作成され、その後、ScrolledWindow の XmNworkWindow のリソースがグラフにセットされます。

```
Widget scrollw, graph;
scrollw = XtVaCreateManagedWidget("scrollw",
  xmScrolledWindowWidgetClass, toplevel,
  XmNscrollingPolicy, XmAUTOMATIC,
  NULL);
graph = XtVaCreateManagedWidget("my_graph",
  xtXrtGraphWidgetClass, scrollw,
  XtNwidth, 900, /* Set graph size here */
  XtNheight, 400,
  NULL);
XtVaSetValues(scrollw,
  XmNworkWindow, graph,
  NULL);
```

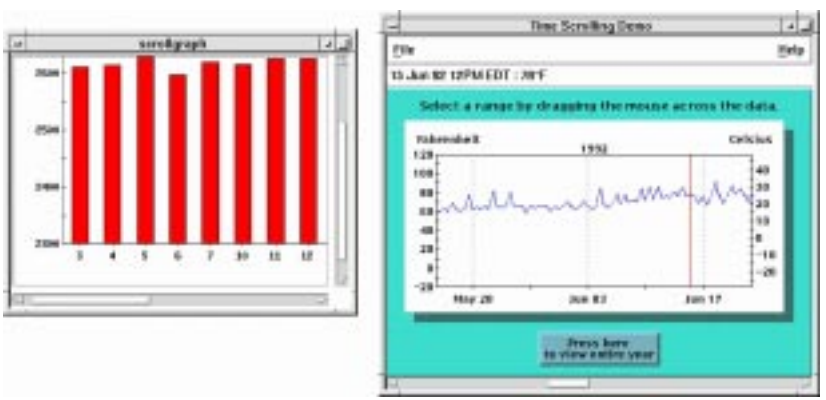


図62　スクロールバーが表示されているグラフ

スクロールするグラフの例(*scrollgraph.c*)は、*$XRTYOME/src/graph/example* にあります。

グラフ全体を表示するより、スクロールバーを使用してグラフに表示されるデータをコントロールする方が良い場合があります。*$XRTHOME/src/graph/demos/scroll_time* は、その例を示していますので参照してください。

4.20 出力と印刷

多くのアプリケーションでは、ユーザにグラフのハードコピーを提供する必要があります。XRT/graph は、グラフをファイルやプリンタに出力する機能を備えています。

XRT/graph の印刷方法は、グラフのウィンドウが他のウィンドウで隠されていても正常に働きます。各方法は下記に解説されていますが、詳細は Appendix D にありますので参照してください。

XrtDrawPS()

グラフの PostScript 記述を作成します。このルーチンは、ネイティブの PostScript オペレータを使用して Encapsulated PostScript ファイルを作成します。フォントの選択、色の操作、グラフのサイズや位置決めのパラメータがあります。

XrtOutputPS()

*XrtDrawPS()*と同様に、*XrtOutputPS()*はグラフの Encapsulated PostScript 記述を作成しますが、ネイティブの PostScript オペレータを使用しません。その代わりに、イメージまたは colorimage オペレータを使用し、画面のイメージと同等の PostScript bitmap を作成します。このプリント・ファイルはかなり大きく、プリンタの解像度を最大限に利用することができないことがあります。

XrtOutputRaster()

*XrtOutputRaster()*は、グラフの Sun rasterfile フォーマットを作成します。これは、機種が SPARC の場合にのみ使用できます。

XrtOutputXwd()

*XrtOutputXwd()*は、グラフの X Window Dump フォーマットを作成します。

XrtGetImage()

*XrtGetImage()*は、グラフの XImage のポインタを返します。特別な出力フォーマットを実行したい場合に使用できます。

注意: *XrtSaveImageGeneric()*などのイメージに関するルーチンを使用して、グラフは、JPEG、PNG や任意のフォーマットに出力することができます。詳細は、P143 の”XrtPDSImage”を参照してください。

印刷ポップアップ・ウィンドウ

*XrtDrawPS()*の使用法とファイルの *lp* へのパイプの方法を紹介する、印刷のポップアップ・ウィンドウのソースコードが、*$XRTHOME/src/tools/pdbox_graph.c* にありますので参照してください。

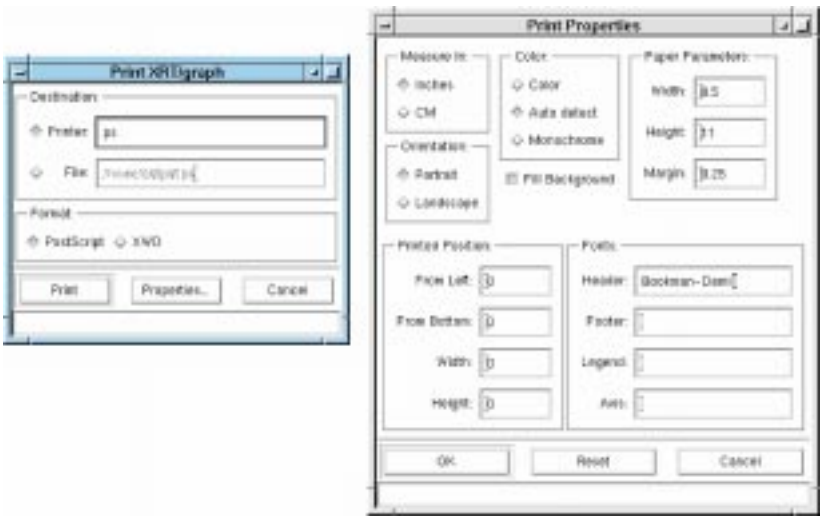


図63　サンプル・コードで作成された印刷のポップアップ

5

XRT/graph Data

グラフへデータを取り込む ■ データ・オブジェクト

オブジェクトとデータ・アクセス法 ■ データの除外と非表示

データのトランスポーズ ■ 複数のデータ・セット

静的データを使用した例 ■ パイプからリアルタイム・データを読み込む例

5.1 グラフへデータを取り込む

グラフ化されるデータは、Unix ファイル、データベース、リアルタイム・データのフィード、または他のマシンで実行されている処理等、あらゆるソースから持ってくることができます。

XRT/graph の開発において、グラフ化するデータの読み込みと、グラフ化するためにそのデータをデータ・オブジェクトに挿入することが最も難しい作業かもしれません。この章では、データ・オブジェクトを解説し、このオブジェクトの割り当てとロードの例を幾つか紹介します。

ファイルのデータ

データが、(リアルタイムに変更またはアップデートしない)Unix ファイルにある場合は、二つの方法が考えられます。ひとつは、ファイルが *XrtDataCreateFromFile()*が理解できる構文に従うようデータを操作する方法です。この手続きは、データ・オブジェクトを割り当て、指定されたファイルのデータをロードし、そのデータのハンドルを返します。*XrtDataCreateFromFile()*とその構文ついては、Appendix D を参照してください。

もう一つは、*XrtDataCreate()*を用いてデータ・オブジェクトを割り当て、(*fgets(3)*か *fscanf(3)*によって)データ・ファイルから読み込まれたデータでセットする方法です。このオブジェクトがロードされると、グラフが作成され、`XtNxrtData` リソースは、データ・オブジェクトを参照する `XrtDataHandle` にセットされます。

リアルタイム・データ

データがリアルタイムに変化し、それに従ってアップデートされるグラフの場合は注意が必要です。Xt Intrinsics には、リアルタイム・イベントに対応するよう、アプリケーションのコールバック・ルーチンを呼び出す機能が提供されています。

*XtAppAddInput()*は、ファイル記述子にペンディングされている入力がある時に呼び出される、アプリケーションのコールバック・ルーチンの登録に使用されます。ファイル記述子は、ファイル、パイプ、または他のマシンで実行されている処理のソケット・コネクションとすることができます。

他には、アプリケーションが *XtAddTimeOut()*を使用して、インターバル・タイマーを定義する方法があります。指定されたインターバルが切れる度に、アプリケーションのコールバック・ルーチンが実行されます。この時、アプリケーションはグラフ化するデータが他にないかを調べ、その結果に従って処理を行うことができます。

これらのルーチンの詳しい情報については、Xt Intrinsics の資料をご覧ください。

高速アップデート

データが連続して加えられていくストリップ・チャートや、新しいポイントを受け取るスキャッター・プロット等を作成するリアルタイムのアプリケーションには、高速アップデート機能を使用することができます。詳細は 7.6 に解説してあります。

5.2 データ・オブジェクト

XRT/graph がグラフを描画する時は、`XtNxrtData` で指されているデータ・オブジェクトのデータ値を使用します。[1]

`XrtDataHandle` は、データ・オブジェクトのハンドルです。データ・オブジェクトのデータや属性は、XRT/graph から提供されている方法を使用して変更することができます。これらの方法は、`XrtDataHandle` を使用してデータ・オブジェクトを参照します。データ・オブジェクトを使用することにより、今後の内部データ構造体の変更からアプリケーションを独立させ、パフォーマンスの下落やメモリに対する要求の増加を防ぐことができます。

primary(一番目の)データ・セット

XRT/graph では、`XtNxrtData2` を使用して、二番目のデータ・オブジェクトも同じグラフに表示させることができます。但し、`XtNxrtData` が `NULL` で、`XtNxrtData2` が有効な `XrtDataHandle` を指している時は、`XtNxrtData2` で参照されているデータが *primary* データ・セットとして使用されます。(この場合、Y2 が primary Y 軸なので)Y2 軸のリソースが、`XtNxrtType2`、`XtNxrtSetLabels2`、`XtNxrtPointLabels2`、`XtNxrtDataStyles2` と共に使用されます。

Data の役割

必要なデータ・オブジェクトを割り当てたり、破棄するのはアプリケーションの役割です。XRT/graph はデータのコピーを作成しませんが、`XtNxrtData` で指されているアプリケーションのメモリ上のデータを参照します。XRT/graph がグラフのウィンドウをリペイント(再描画)する場合は、`XtNxrtData` が指しているデータを使用します。

---

[1] このリソースは、XRT/graph の前のバージョンと互換性を持たせるため、`XrtData` 構造体を指すようにセットすることもできます。但し、データ・オブジェクトの属性の中には、現在のバージョンの API を用いて `XrtData` 構造体にセットできないものもあります。

5.2.1 Array と General データのフォーマット

データ・オブジェクトのデータには、*Array(配列)*と *General(一般)*の二つのフォーマットがあります。

Array(配列)データ

Array データは、X 値を共有する Y 値のセットで定義されます。全てのセットは、同じ数のポイントを持っています。X 値と Y 値は、どちらも `float` 型です。棒、積み上げ棒、円、HiLo、HiLoOpenClose、キャンドル、レーダー、filled レーダーを始め、共通の X 値を持つグラフには、Array データが使用できます。

Array — 2 Sets of 3 Points:

	Point 0	Point 1	Point 2
Common X-values	1	2	3
Set 0 Y-values	150	175	160
Set 1 Y-values	125	100	200

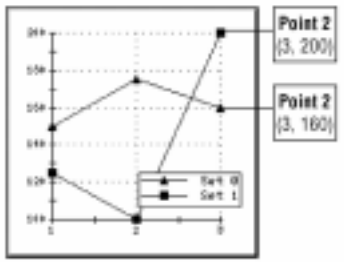


図64　Array データのデータとグラフ

General(一般)データ

General データは、X 値と Y 値のセットで定義されます。各セットは、それぞれ異なる数のポイントを持ちます。X 値と Y 値は、どちらも `float` 型です。General データは、線、面、ポーラー・グラフ、バブル・チャートにのみ使用できます。

注意: logical set 内の各セットは、必ず同じ数のポイントを持つようにしてください。

General — 2 Sets, Maximum of 4 Points:

		Point 0	Point 1	Point 2	Point 3
Set 0 4 Points	X-values	1	1.25	1.75	3
	Y-values	120	140	160	170
Set 1 2 Points	X-values	1.5	2.5		
	Y-values	130	190		

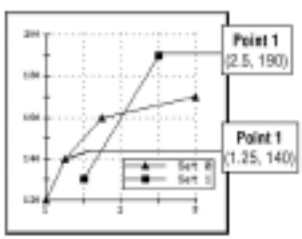


図65　General データのデータとグラフ

5.2.2 XrtDataHandle の属性

データ・オブジェクトは、グラフのデータを持つ以外に、次の変更可能な属性を持っています。

Type　データが Array か General かを指定。*XrtDataGetType()*は、現在のフォーマットを取得する。Array から General に変更するには、*XrtDataSetType()*を使用。General から Array には変更不可。

Nsets　データ・セット内のセット数を指定。*XrtDataGetNSets()*を使用して現在の値を取得し、*XrtDataSetNSets()*でそれを変更できる。

Npoints　Array のすべてのセットまたは、General の一つのセットの、合計ポイント数を指定。*XrtDataGetNPoints()* を使用して現在の値を取得し、*XrtDataSetNPoits()*でそれを変更できる。

Hole　ある Y 値を hole(欠けているデータ)と指定する。デフォルトの hole 値は、XRT_HUGE_VAL 。*XrtDataGetHole()* を使用して現在の値を取得し、*XrtDataSetHole()*でそれを変更できる。

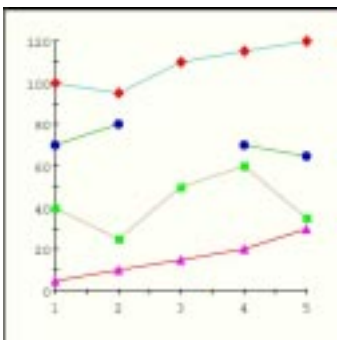


図 66　データの hole

5.3 オブジェクトとデータ・アクセス法

XRT/graph には、データ・オブジェクト内のデータの設定や取得を行う手続きや方法、またはデータ・オブジェクト自身を操作する方法が幾つかあります。これらは、下の表に一覧がありますが、Appendix D にも解説されています。簡易手続きの多くは、ソースコード(*$XRTYOME/src/graph/util*)でも提供されています。

セット/ポイントの追加と削除	
XrtArrDataAppendPts()	ポイントの列を Array データ・オブジェクトに追加する
XrtGenDataAppendPt()	(x, y)値を General データ・オブジェクトのデータの最後に追加する

XrtArrDataRemovePts()	Array データ・オブジェクトの各セットからポイントを削除する
XrtGenDataRemovePt()	General データ・オブジェクトからポイントを削除する
XrtDataRemoveSet()	データ・オブジェクトからセットを削除する
	データ値の取得と変更
XrtDataGetXData() *XrtDataGetYData()*	セットの X 値または Y 値の内部リストのポイントを取得する *XrtDataRelease()*が呼ばれるまで、他のデータ変更はできない
XrtDataGetXElement() *XrtDataGetYElement()*	セットとポイント・インデックスの X 値または Y 値を返す
XrtDataRelease()	データ・オブジェクトのコントロールを XRT/graph に返す
XrtDataSetXData() *XrtDataSetYData()*	セットの X 値または Y 値のリストをセットする
XrtDataSetXElement() *XrtDataSetYElement()*	セットとポイント・インデックスの X 値または Y 値をセットする
XrtDataSort()	X 値の増加に従って、各セットのポイントをソートする
XrtDataUpdateDataValue()	データ・オブジェクトのデータ値を新しい値へ変更する
	データ・オブジェクトの操作
XrtDataConcat()	二つの既存のデータ・オブジェクトによって指されているデータを結合させて新しいデータ・オブジェクトを作成する
XrtDataConvertToHandle()	以前のバージョンで使用されていた XrtData 構造体をデータ・オブジェクトに変換する
XrtDataCopy()	データ・オブジェクトのコピーを作成する
XrtDataCreate()	新しいデータ・オブジェクトを作成する
XrtDataCreateFromFile()	ファイルのデータで新しいデータ・オブジェクトを作成する
XrtDataDestroy()	データ・オブジェクトを破棄する
XrtDataExtractSet()	既存のデータ・オブジェクトの一つのデータ・セットがロードされた、新しいデータ・オブジェクトを作成する
XrtDataSaveToFile()	データ・オブジェクトのデータをファイルに書く
XrtDataTranspose()	データ・オブジェクトのセットとポイントをトランスポーズ(入れ替え)する

次のコードは、これらの方法を使用して、Array または General データ・オブジェクトのデータ値を出力しています。

```
static void
print_arr(XrtDataHandle d)
{
    /* Print out the data contained in XRT_DATA_ARRAY type data object */
    int i,j;

    printf("\nPrinting Array data with %d sets, %d points\n",
        XrtDataGetNSets(d), XrtDataGetNPoints(d, NULL));
    printf("X-Values:\n");
    for (j = 0; j < XrtDataGetNPoints(d, NULL); j++)
        printf("%f\t", XrtDataGetXElement(d, NULL, j));
    for(i = 0; i < XrtDataGetNSets(d); i++) {
        printf("Y %d Values:\n", i);
        for (j = 0; j < XrtDataGetNPoints(d, NULL); j++)
            printf("%f\t", XrtDataGetYElement(d, i, j) );
        printf("\n");
    }
}

static void
print_gen(XrtDataHandle d)
{
/* Print out the data contained in XRT_DATA_GENERAL type data object */
    int i,j;

    printf("\nPrinting General data with %d sets\n",XrtDataGetNSets(d));
    for(I = 0; i < XrtDataGetNSets(d); i++) {
        printf("Set %d Values:\n", i);
        for (j = 0; j < XrtDataGetNPoints(d, i); j++)
            printf("(%f, %f)\t",
            XrtDataGetXElement(d, i, j), XrtDataGetYElement(d, i, j));
        printf("\n");
    }
}

static void
pr_data(XrtDataHandle d)
{
    if ( XrtDataGetType(d) == XRT_DATA_ARRAY ) /* is array */
        print_arr(d);
    else /* is general */
        print_gen(d);
}
```

図67　データ・オブジェクトの出力ルーチン

データに直接アクセスする

通常、アプリケーションは、データ・オブジェクトに使用されるメモリに直接アクセスすることはありません。ほとんどのデータ・オブジェクトの方法は、渡された値をオブジェクトの内部データ構造体にコピーします。

X 値または Y 値を素早く変更する場合には、特別な方法が必要です。XRT/graph の *XrtDataGetXData()* と *XrtDataGetYData()* は、X 値または Y 値の配列の内部ポインタを返しますので、アプリケーションはこの配列の浮動小数点値を直接変更することができます。

但し、*XrtDataRelease()*を呼び出して、配列の所有権を XRT/graph に返すまでは、オブジェクトに対するその他のデータ変更はできません。次の例では、これらの方法を使用して、セットの X 値と Y 値を増加、減少させています。

```
static void
delta_set(XrtDataHandle d, int set, float xdel, float ydel)
{
    /* Add xdel and ydel to all points in set */
    int i, npoints;
    float *xdata, *ydata;
    if ((npoints = XrtDataGetNPoints(d, set)) < 0) {
      return;
    }

    xdata = XrtDataGetXData(d, set);
    ydata = XrtDataGetYData(d, set);
    if (xdata && ydata) {
      for (i = 0; i < npoints; i++) {
        xdata[i] += xdel;
        ydata[i] += ydel;
      }
    }
    XrtDataRelease(d);

    XtVaSetValues(graph, XtNxrtData, d, NULL);
}
```

5.4 データの除外と非表示

デフォルトでは、データ・オブジェクトに含まれる全てのデータがグラフ化されます。XRT/graph では、データ・オブジェクトの属性を使用することにより、特定なセットまたはポイントの範囲だけをグラフ化したり、グラフから一時的に指定したセットだけを除外したり非表示にしたりすることができます。

セットとポイントの開始と終了の定義

データ・オブジェクトは、サブセットや、オブジェクト内の指定したセットとポイントの範囲だけをグラフ化することができます。この方法を使用して、例えば、データ・オブジェクト内の過去のデータも保持しながら、それらを表示せずに現在のデータだけを表示させることができます。次のデータ・オブジェクトの属性を使用して、表示される最初と最後のセットとポイントを指定することができます。

FirstSet　グラフ化される最初のセットのインデックスを指定。*XrtDataGetFirstSet()*を使用して現在値を取得し、*XrtDataSetFirstSet()*でそれを変更できる。

LastSet　グラフ化される最後のセットのインデックスを指定。*XrtDataGetLastSet()*を使用して現在値を取得し、*XrtDataSetLastSet()*でそれを変更できる。

FirstPoit　Array の全てのセットまたは、General の一つのセットの、グラフ化される最初のポイントのインデックスを指定。*XrtDataGetFirstPoint()*を使用して現在値を取得し、*XrtDataSetFirstPoint()*でそれを変更できる。

LastPoint　Array の全てのセットまたは General の一つのセットの、グラフ化される最後のポイントのインデックスを指定。*XrtDataGetLastPoint()*を使用して現在値を取得し、*XrtDataSetLastPoint()*でそれを変更できる。

個々のセットの除外と非表示

XRT/graph では、グラフの表示から、一時的に一つまたは複数のセットを削除することができます。*XrtDataSetDisplay()*と *XrtDataGetDisplay()*は、データ・オブジェクトの Display 属性にアクセスすることができます。有効な値は次の通りです。

`XRT_DISPLAY_SHOW`　セットはグラフに表示される。デフォルト値。

`XRT_DISPLAY_HIDE`　セットは表示されていないが、グラフの軸やその他の要素の位置決めの計算に使用される。

`XRT_DISPLAY_EXCLUDE`　セットは、グラフから除外される。

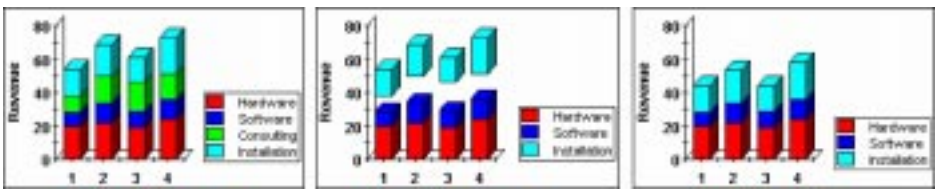


図 68　Display 属性 - _SHOW、_HIDE、_EXCLUDE

HiLo/HiLoOpenClose、キャンドル、バブルの場合の除外と非表示

これらのグラフ・タイプで、logical set の内一つのセットを除外すると、logical set 全体が除外されます。また、high または low のセットのどちらかを非表示にすると logical set 全体も非表示になり、キャンドルの open または close のどちらかのセットを非表示にすると、open、close 両方とも非表示になります。

5.5　データのトランスポーズ

グラフをトランスポーズすると、セットとポイントが入れ替わります。例えば、図 4(P13)にある *mm63.data* には、四つのポイントのセットが二つありますが、これがトランスポーズされると、二つのポイントのセットが四つに変更されます。但し、データ・セットによっては、トランスポーズに向かないものがあります。

トランスポーズには、二つの方法があります。*XrtDataTranspose()*はデータ値をトランスポーズし、*MtGraphTransposeData()*はグラフ全体をトランスポーズ(データのセットとポイント、セット・ラベルとポイント・ラベル、マーカとテキスト領域の位置を全て入れ替え)します。

図 69 は、簡単な棒グラフをトランスポーズした例です。この例では、トランスポーズされたデータの棒グラフは、(各四半期毎の)四つの棒の(Revenue と Expenseses の)グループを二つ表示しています。元のデータでは、(Revenue と Expenses の)二つの棒の(各四半期毎の)グループを四つ表示しています。

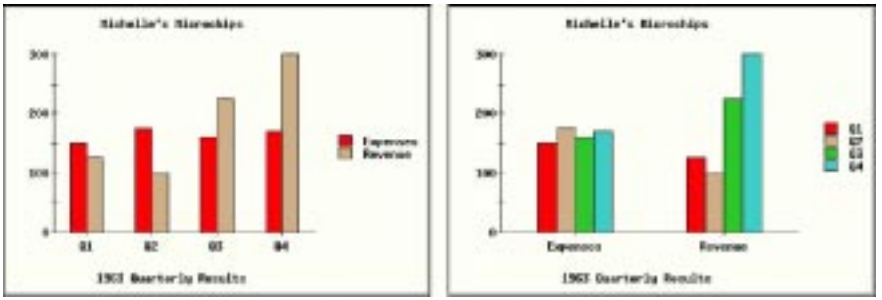


図69　通常(左)とトランスポーズされた(右)グラフ

MtGraphTransposeData()(ソースは、*$XRTHOME/src/tools/mtgraph.c* にあります。)

データ、セットとポイント・ラベル、マーカとテキスト領域の貼り付けをトランスポーズします。この方法を使用するには、*$XRTHOME/src/tools/mtgraph.h* をインクルードし、プログラムを XRT の"tools"ライブラリとリンクしてください。

```
Boolean
MtGraphTransposeData(
  Widget           graph,
  XrtDataHandle    data,
  XrtDataHandle    data2,
  Boolean          sw_labels,
  Boolean          sw_markers,
  Boolean          sw_text
  )
```

data は(既にトランスポーズされている)新しいデータ・オブジェクトで、`NULL` の場合は、ウィジェットの現在のデータがトランスポーズされ、`NULL` でない場合は、渡されたオブジェクトをウィジェットにセットします。*data2* は、複合グラフの場合の二番目のデータセット。*sw_labels* は、ポイント・ラベルとセット・ラベルかを入れ替えるかどうか、*sw_markers* は、`XtNxrtXMarkerPoint` と `XtNxrtXMarkerSet` の値を入れ替えるかどうか、*sw_text* は、`XRT_TEXT_ATTACH_DATA` によって貼り付けられているテキスト領域のセットとポイントの要素を入れ替えるかどうかをそれぞれ指定します。この手続きは成功時には `True`、失敗時には `False` を返します。

*XrtDataTranspose()*は、Appendix D にも解説されています。

5.6　複数のデータ・セット

XRT/graph では、グラフの属性は変更せずに、データ・セット全体を変更することができます。プログラムでは、複数のデータ・オブジェクトを作成し、`XtNxrtData` または `XtNxrtData2` を現在のデータを指すようにセットして、それらを切り替えることができます。

図 70 のコードは、`XtNxrtData` が該当するデータを指すようにセットし、ヘッダー・テキストをアップデートしています。

```
static char *dec_st[] = { "DEC", NULL, };

void
dec_proc(w, client_data, call_data)
Widget w;
XtPointer client_data;
XmToggleButtonCallbackStruct *call_data;
{
   if (call_data->set)
       XtVaSetValues(graph,
            XtNxrtData, dec0990,
            XtNxrtHeaderStrings,dec_st,
            NULL);
}
```

図70　コールバック手続きでデータ・セットを変更する

データ変更を示す例が、*stock.c*(*$XRTHOME/src/graph/example*)にありますので参照してください。

リアルタイムでも、データは簡単に変更することができます。最初に、新しいデータ・オブジェクトを作成し、新しいデータでセットするか、または、現在表示されているデータをアップデートします。次に、XtNxrtData をセットして、グラフに新しいデータを使用するように指示してください。

5.7　静的データを使用した例

コンパイル時にグラフ化されるデータが明らかな場合は、それを静的変数で定義し、データ・オブジェクトへコピーすることができます。

次の例は、表示される値を静的 C 変数の xdata[]、ydata[]に定義しています。プログラムは正しいサイズのデータ・オブジェクトを割り当て、データ・オブジェクトの方法を使用して静的値を構造体へコピーしています。

```
#define NSETS 3
#define NPOINTS 5
static float xdata[NPOINTS] = {1, 2, 3, 4, 5,}; /* X-values */
static float ydata[NSETS][NPOINTS] = {
    {5, 8, 14, 19, 24},           /* Mexico */
    {11, 15, 18, 21, 22},     /* Tokyo */
    {14, 16, 15.5, 15.2, 15.4},   /* New York */
};
static char *sl[] = { "Mexico", "Tokyo", "New York", NULL,};
static char *pl[] = { "60", "70", "80", "90", "2000", NULL,};
...
Widget graph, toplevel;
XrtDataHandle pop;
int i;
/* Allocate and load the data */
pop = XrtDataCreate(XRT_DATA_ARRAY, NSETS, NPOINTS);
for (i=0; i<NPOINTS; i++) {
    XrtDataSetXElement(pop, NULL, i, xdata[i]);
}
for (i=0; i<NSETS; i++) {
    XrtDataSetYData(pop, i, ydata[i], NPOINTS, 0);
}
...
/* Create graph */
graph = XtVaCreateManagedWidget("mygraph",
    xtXrtGraphWidgetClass, toplevel,
    XtNxrtData, pop,
    XtNxrtSetLabels, sl,
    XtNxrtPointLabels, pl,
    XtNxrtXAnnotationMethod, XRT_ANNO_POINT_LABELS,
    NULL);
```

図71　静的C変数からデータ･オブジェクトをロードする

5.8　パイプからリアルタイム･データを読み込む例

vmgraph デモ･プログラムは、パイプからリアルタイム・データを読み込み、入ってくるデータを分析、グラフ化します。*vmgraph* のソースコードは*$XRTHOME/src/graph/demos/vmgraph* ディレクトリにあります。

三つのグラフ･オブジェクトを作成した後、*vmgraph* は子プロセスを分岐し、それが、*vmstat(8)* を開始します。子プロセスの標準出力は、*vmgraph* プロセスの標準入力(ファイル記述子は 0)になるようにセットアップされます。*vmgraph* は次に、*XtAddInput()*を用いて、ファイル記述子 0 にペンディングされている入力がある時にはいつも *read_line()*を呼ぶように、Xt Intrinsics に命令します。

```
XtAddInput(0、XtInputReadMask、read_line、NULL);
```

子プロセスを分岐し、パイプ･コミュニケーションをセットアップするコードは以下のようになります。

```
if (pipe(pipes) == -1) {
    fprintf(stderr, "pipe failed\n");
    exit(1);
}
switch(vmpid = fork()) {
    case -1:
        fprintf(stderr, "fork failed\n");
        exit(1);
    case 0: /* child, vmstat writes to pipe */
        close(pipes[0]);
        dup2(pipes[1], 1);
        sprintf(refr, "%d", sample_time);
        execl("/usr/ucb/vmstat", "vmstat", refr, 0);
        fprintf(stderr, "execl failed!\n");
        break;
    default: /* parent reads from pipe, waits for child */
        close(pipes[1]);
        dup2(pipes[0], 0);
        XtAppAddInput(app_context, 0, XtInputReadMask,
            read_line, NULL)
        wait3(NULL, WNOHANG, NULL);
        break;
}
```

図72　プロセスを分岐しパイプを通して読み込む

*read_line()*ルーチンは、*vmstat* からデータを一行読み込みます。それを個々のデータ値に分析した後、アプリケーションのグラフ・オブジェクトをアップデートするルーチンを呼び出します。

6

ユーザ･インタラクションのプログラミング

デフォルトのユーザ･インタラクション ■ トランスレーションとアクションの概要
ユーザ･インタラクションの開始 ■ ユーザ･インタラクションのアップデート
ユーザ･インタラクションの終了 ■ グラフ･データとのインタラクション
ウィンドウのリサイズ

この章では、XRT/graph のユーザ･インタラクションとそのコントロール法について解説します。

6.1 デフォルトのユーザ･インタラクション

デフォルトでは、全てのグラフでスケーリング、移動、ズームを行うことができます。また、3D 表示になっている棒グラフ、積上げ棒グラフ、円グラフは、ユーザがインタラクティブに回転することができます。[1] 図 73 は、デフォルトで実行できるインタラクションの一覧です。

プロパティ･エディタ

マウスの右ボタンで XRT/graph のプロパティ･エディタを呼び出すトランスレーションをインストールすることができます。但し、プロパティ･エディタはアプリケーションで使用できるようにセットされていなければ表示されません。(3.1 参照)

6.2 トランスレーションとアクションの概要

XRT/graph のデフォルトのトランスレーション･テーブルは、インタラクティブなアクションを呼び出すユーザ･イベントを定義しています。このユーザ･インタラクション機能は、次のメカニズムを利用してカスタマイズすることができます。

[1] 3D 表示のグラフを回転するには、奥行き(`XtNxrtGraphDepth`)が 0 より大きく設定されている必要があります。

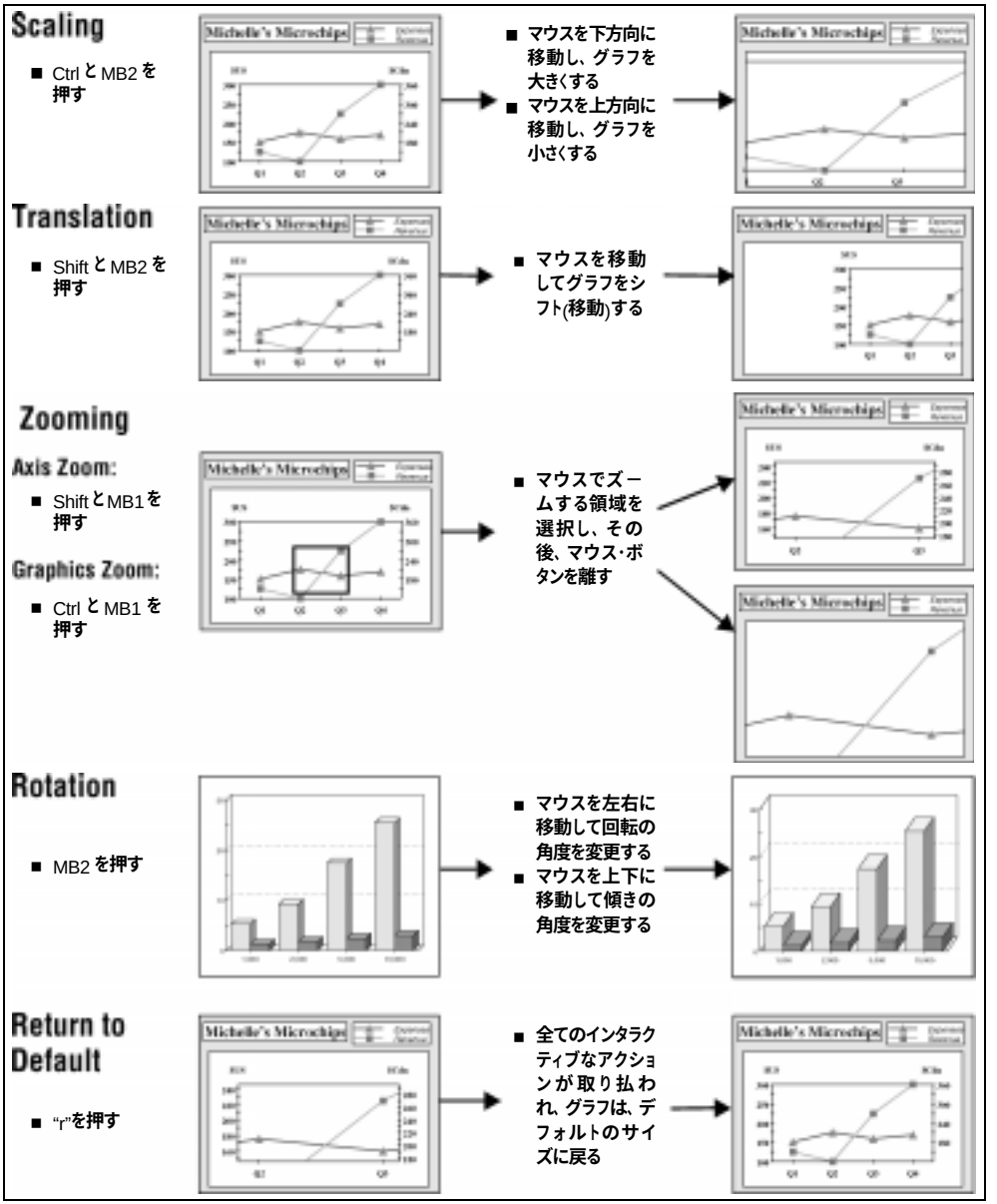


図73　デフォルトのユーザ・インタラクションのトランスレーション

- **トランスレーション･テーブル** - アプリケーションは、独自のトランスレーションをウィジェットのトランスレーション･テーブルに追加して、デフォルトのトランスレーションを置き換えることができます。特定なインタラクションを不可に設定したり、インタラクションを呼び出すマウスやキーボードを変更する時に、この方法が使用できます。Appendix E は、XRT/graph のデフォルトのトランスレーションとアクションの一覧がありますので参照してください。
- **コールバック** - ユーザ･インタラクションの前、最中、後に呼び出されるコールバック手続きを登録することにより、アプリケーションは、ユーザがウィジェットとインタラクションを行う度に通知を受けることができます。あるインタラクションを不可に設定したり、制限したりする時に、この方法が使用できます。ここから先のセクションでは、各インタラクションのコールバックの使用法が解説されています。

インタラクションの三つの段階

ユーザ･インタラクションは、次の三つの段階で処理されます。

1 ユーザ･インタラクションの開始
2 ユーザ･インタラクションのアップデート
3 ユーザ･インタラクションの終了

一つのインタラクションは、必ずこの段階を順番に通ります。アプリケーションは各段階でコールバックによる通知を受けることができますので、これを利用してあるインタラクションを不可に設定したり、アクションを変更したりすることができます。

6.3 ユーザ･インタラクションの開始

*ModifyStart()*が呼び出され、ユーザ･インタラクションが開始されることをアプリケーションに通知します。図 74 は、このアクション･ルーチンの図解で、追加された Modify コールバックがどこで呼び出されるかを示しています。

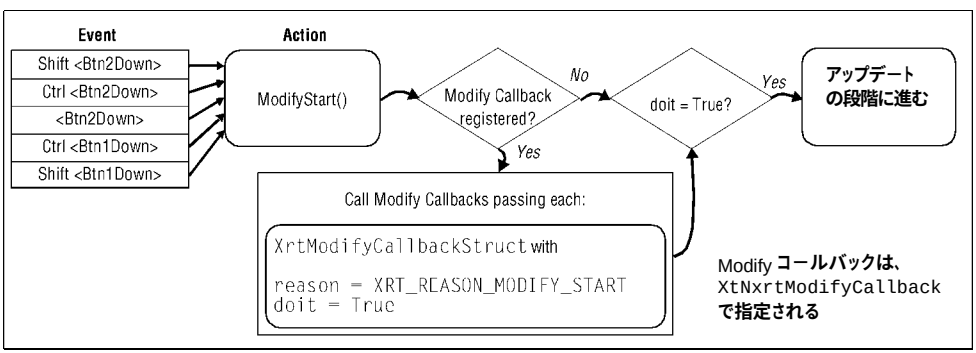


図74 ModifyStart()アクション･ルーチン

全てのユーザ･インタラクションを不可にする

インストールされているトランスレーション･テーブルに関らず、XtNxrtModifyCallback を使用してユーザ･インタラクションを不可に設定することができます。各コールバック手続きには、次の構造体のポインタが渡されます。

```
typedef struct {
  int reason; // Read-only
  XEvent *event; // Read-only
  String *params; // Read-only
  Cardinal nparams; // Read-only
  Boolean doit;
} XrtModifyCallbackStruct;
```

全てのユーザ･インタラクションを不可にするには、(*reason* パラメータが XRT_REASON_MODIFY_START の時)この構造体の *doit* パラメータを False にしてください。*doit* が False の時は、*ModifyStart()*は、次の *ModifyStart()*が呼び出されるまで、全てのアップデートのアクションを不可にします。

6.4 ユーザ･インタラクションのアップデート

幾つかのアクション･ルーチンの内の一つが呼び出され、ユーザ･インタラクションがアップデートされることをアプリケーションに通知します。インタラションが *ModifyStart()* アクションを通るまで、これらのルーチンはグラフをアップデートしません。

6.4.1 スケーリング、移動、ズーム

*ScaleUpdate()*はグラフのインタラクティブなスケーリング、*TranslateUpdate()*はインタラクティブな移動をそれぞれアップデートします。*ZoomEnd()*は、(*ZoomUpdate()*で定義される)"zoom rubberband"で指定された箇所をズームします。これらのルーチンは、全て Graph Margin リソース[1] を変更します。図 75 は、これらのアクション･ルーチンの図解で、追加した Transform と Expose コールバックがどこで呼ばれるかを示しています。

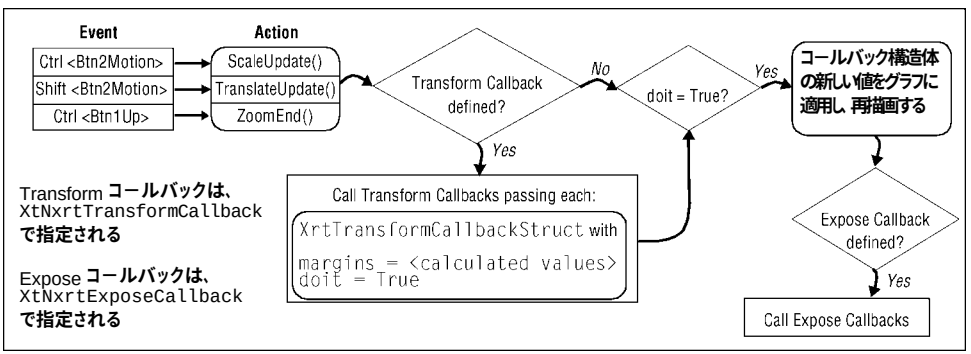


図 75　ScaleUpdate()、TranslateUpdate()、ZoomEnd()アクション･ルーチン

[1] Graph Margin リソースは、XtNxrtGraphMarginBottom、XtNxrtGraphMarginTop、XtNxrtGraphMarginLeft、XtNxrtGraphMarginRight です。

インタラクションのコントロール

スケーリング、移動、ズームは、XtNxrtTransformCallback でコントロールできます。各コールバックには、次の構造体のポインタが渡されます。

```
typedef struct {
  int          reason;      // Read-only
  XEvent       *event;      // Read-only
  String       *params;     // Read-only
  Cardinalnparams;     // Read-only
  Boolean      reset;       // Read-only
  Positionleft_margin;
  Positionright_margin;
  Positiontop_margin;
  Positionbottom_margin;
  Boolean      doit;
} XrtTransformCallbackStruct;
```

Transform コールバックは、*left_margin*、*right_margin*、*top_margin*、*bottom_margin* と *doit* パラメータを変更します。*ScaleUpdate()*、*TranslateUpdate()*、*ZoomEnd()*は、これらのパラメータをグラフに適用します。図 76 では、Transform コールバックのマージンのパラメータを調べ、変更し、スケーリングを制限しています。

```
void
limit(Widget mygraph, XtPointer cd, XtPointer cbs)
{
    XrtTransformCallbackStruct *tcbs=(XrtTransformCallbackStruct *)cbs;

if (! cbs->reset){
        if (cbs->left_margin < -100)cbs->left_margin = -100;
        if (cbs->left_margin > 100)cbs->left_margin = 100;
        if (cbs->right_margin < -100)cbs->right_margin = -100;
        if (cbs->right_margin > 100)cbs->right_margin = 100;
        if (cbs->top_margin < -100)cbs->top_margin = -100;
        if (cbs->top_margin > 100)cbs->top_margin = 100;
        if (cbs->bottom_margin < -100)cbs->bottom_margin = -100;
        if (cbs->bottom_margin > 100)cbs->bottom_margin = 100;
    }
}
```

図 76　Margin を±100 に制限するコールバック

上のコールバック・ルーチンを登録するには、次のコールを使用してください。

```
XtAddCallback(mygraph,
  XtNxrtTransformCallback, Limit, Null);
```

6.4.2　軸のズーム

*ZoomAxisEnd()*は、軸を表示したままグラフをズームします。ズームされる領域は、(*ZoomAxisUpdate()*で作成される)"zoom rubberband"で定義されます。このルーチンは、XtNxrt[XYY2]AxisMax と XtNxrt[XYY2]AxisMin リソースを変更します。図 77 は、このアクション・ルーチンの図解で、追加した ZoomAxis と Expose コールバックがどこで呼ばれるかを示しています。

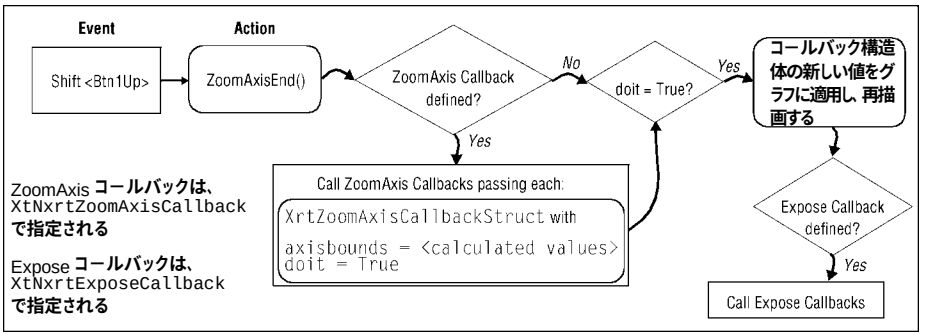


図 77　ZoomAxisEnd()アクション・ルーチン

インタラクションのコントロール

XtNxrtZoomAxisCallback を使用して軸のズームをコントロールすることができます。各コールバックには次の構造体のポインタ渡されます。

```
typedef struct {
  int          reason; // Read-only
  XEvent       *event; // Read-only
  String       *params; // Read-only
  Cardinalnparams; // Read-only
  Boolean      reset;  // Read-only
  float        new_xaxis_min;
  float        new_xaxis_max;
  float        new_yaxis_min;
  float        new_yaxis_max;
  float        new_y2axis_min;
  float        new_y2axis_max;
  Boolean      doit;
} XrtZoomAxisCallbackStruct;
```

ZoomAxis のコールバックは、*new_xaxis_min*、*new_xaxis_max*、*new_yaxis_min*、*new_yaxis_max*、*new_y2axis_min*、*new_y2axis_max* または *doit* パラメータを変更することができます。*ZoomAxisEnd()*は、これらのパラメータをグラフに適用します。例えば、ズームをある軸の範囲に制限するには、ZoomAxis コールバックの軸のパラメータを調べて変更してください。

6.4.3　スケーリング、移動、ズームのリセット

*ViewReset()*アクションは、Margin と Axis の min/max(最小と最大)の UseDefault リソースを True にセットして、インタラクティブなスケーリング、移動、ズームをリセットします。但し、ウィジェットに対するユーザ・インタラクションが行われなければ、このアクションは、これらのリソースをリセットしません。

*ViewReset()*は、登録されている Transform コールバックを開始し、*reset* と *doit* が True にセットされた XrtTransformCallbackStruct のポインタを渡します。このアクションは、次に登録されている ZoomAxis コールバックを開始し、*reset* と *doit* が True にセットされた XrtZoomAxisCallbackStruct のポインタを渡します。

アプリケーションは、Transform または ZoomAxis のコールバックを使用して、Graph Margin または Axis のリセットを拒否することができます。どちらのコールバックも、リセットを拒否するには、*reason* が `True` の時にコールバックの *doit* パラメータを `False` に変更してください。

*ViewReset()*は、登録されている Expose コールバックも開始します。

6.4.4 回転

*RotateUpdate()*アクションは、グラフのインタラクティブな回転をアップデートします。`XtNxrtGraphDepth` がゼロより大きい場合の 3D 表示の棒、積み上げ棒、円グラフでのみインタラクティブな回転が実行できます。このルーチンは、`XtNxrtGraphRotation` と `XtNxrtGraphInclination` リソースを変更します。
図 78 は、このアクション・ルーチンの図解で、追加した Rotate と Expose のコールバックがどこで呼ばれるかを示します。

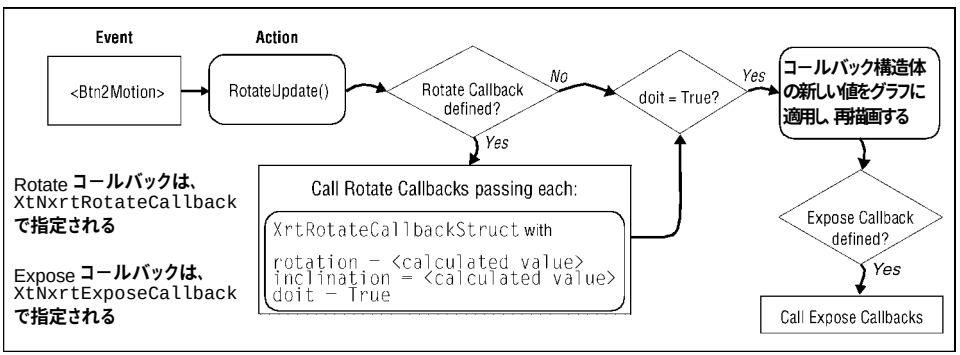


図 78　RotateUpdate()アクション・ルーチン

回転のコントロール

`XtNxrtRotateCallback` を使用して回転をコントロールすることができます。各コールバックには次の構造体のポインタが渡されます。

```
typedef struct {
  int          reason; // Read-only
  XEvent       *event; // Read-only
  String       *params; // Read-only
  Cardinalnparams; // Read-only
  int          rotation;
  int          inclination;
  Boolean      doit;
} XrtRotateCallbackStruct;
```

Rotate コールバックは、*rotation*、*inclination*、*doit* パラメータを変更します。*RotateUpdate()*は、これらのパラメータをグラフに適用します。例えば、傾きを 15 から 30 に制限するには、Rotate コールバックの *inclination* パラメータを調べ変更します。

6.4.5 インタラクションの不許可

特定なユーザ・インタラクションを不可にするには、そのトランスレーションを *NoAction()*にセットする方法が一番簡単です。6.3 には、全てのインタラクションを不可にする方法が解説してありますので参照してください。次のコードは、各インタラクションを不可にしています。

```
*.XtXrtGraph.translations: #override \n\
  Ctrl         <Btn1Motion>:NoAction() \n\
  ~Ctrl ~Shift <Btn2Motion>:NoAction() \n\
  Shift            <Btn2Motion>:NoAction() \n\
  Ctrl         <Btn2Motion>:NoAction() \n\
  <Key>r:                           NoAction()
```

場合によっては、コールバックを使用してインタラクションのアップデートを不可にすることができます。コールバックで、該当するコールバック構造体の *doit* パラメータを `False` にセットしてください。コールバックを開始したアクション・ルーチンはこのパラメータを調べ、*doit* が `False` であれば変更をグラフに適用しません。(図 74 から図 78 参照)。但し、*ModifyEnd()*によって開始された Modify コールバックは、アプリケーションにユーザ・インタラクションが終了されたことを通知するだけですので、例外です。

次のコールバック手続きは、`XrtTransformCallbackStruct` の *doit* パラメータを `False` にセットして、ユーザがグラフをリセットできないようにしています。

```
void
deny_reset(Widget w, XtPointer cd, XtPointer cbs)
{
  XrtTransformCallbackStruct *tcbs =
    (XrtTransformCallbackStruct *)cbs;
  if (cbs->reset){
    cbs->doit = False;
  }
}
```

上のコールバック・ルーチンを登録するには、次のコールを使用してください。

```
XtAddCallback(mygraph,
    XtNxrtTransformCallback, deny_reset, NULL);
```

6.5 ユーザ・インタラクションの終了

*ModifyEnd()*アクションが呼ばれ、ユーザ・インタラクションが終了したことをアプリケーションに通知します。アプリケーションは、`XtNxrtModifyCallback` を使用してこの通知を受けることができます。各コールバック手続きには、*reason* パラメータが `XRT_REASON_MODIFY_END` にセットされた `XrtModifyCallbackStruct` 構造体のポインタが渡されます。この構造体内のパラメータの変更は意味を持ちません。

6.6 グラフ･データとのインタラクション

グラフ･データのユーザ･インタラクションを操作するには、アクション･ルーチン/コールバックとイベント･ハンドリング手続きの二つの方法があります。通常、アクション･ルーチンを使用する方が簡単です。

6.6.1 アクション･ルーチンとコールバック

ユーザがグラフ･データとインタラクションを行おうとする時にアプリケーションに通知し、クリックされたグラフ座標またはユーザ･イベントに一番近いデータ値を受け取るには、トランスレーション･テーブルを使用できます。XRT/graph には、*Pick()*と *Map()*という二つのアクション･ルーチンが提供されています。これらのルーチンを使用するには、次のようにしてください。

- ユーザ･イベントをアクション･ルーチンへマップするトランスレーションを設定。(どちらのルーチンにもデフォルトのトランスレーションがありません。)
- Pick または Map コールバックを登録。これらは、*Map()*または *Pick()*アクションの最中に開始され、*XrtPick()*と *XrtMap()*の結果が渡されます。

Appenedix E に、XRT/graph のアクション･ルーチンの詳細が解説されています。

Pick()アクション

*Pick()*は、ユーザ･イベントの位置に一番近いデータ値を決定し、それをコールバック手続きに渡します。*Pick()*は、*XrtPick()*手続きを使用します。この *Pick()*を用いて、ユーザがクリックした箇所に一番近いセットとポイントを見つけることができます。(*XrtPick()*の詳細は P120 を参照してください。)図 79 はこのアクション･ルーチンの図解で、どこで Pick コールバックが開始されるかを示します。

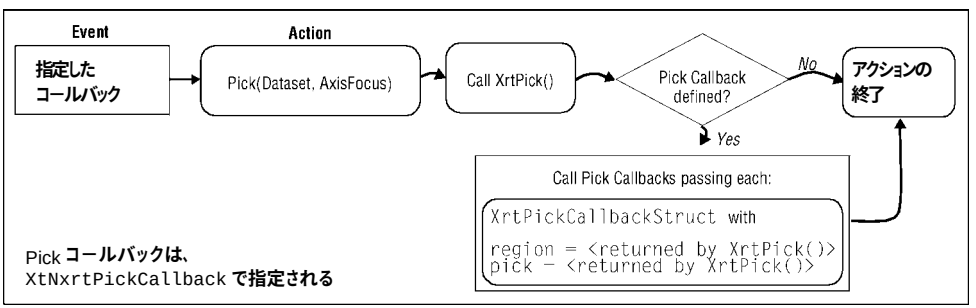


図 79　Pick()アクション･ルーチン

*Pick()*が呼び出されるためには、下記のようにしてイベントに割り当ててください。

```
*.XtXrtGraph.translations: #override \n\
  ~Shift ~Ctrl <Btn1Down>: Pick()
```

*Pick()*は、*XrtPick()*の呼び出しに使用される、データ･セットと軸のフォーカスの二つのオプションのパラメータを取ります。デフォルトでは、*Pick()*は、全てのデータと両方の軸を使用します。

*XrtPick()*の結果を受け取るには、XtNxrtPickCallback を使用してコールバックを登録してください。各コールバックには、次の構造体のポインタが渡されます。

```
typedef struct {
  int              reason;  /* Read-only */
  XEvent           *event;  /* Read-only */
  String           *params; /* Read-only */
  Cardinal     nparams; /* Read-only */
  XrtRegion        region;  /* Read-only */
  XrtPickResult    pick;    /* Read-only */
} XrtPickCallbackStruct;
```

この構造体のフィールドの変更は意味を持ちません。

Map()アクション

*Map()*は、ユーザ・イベントの pixel 座標にマップされたグラフ座標を決定し、それをコールバック手続きに渡します。*Map()*は、*XrtMap()*手続きを使用します。この *Map()*を使用して、どの値(x, y)をユーザがクリックしたかを見つけることができます。(*XrtMap()*の詳細は P123 を参照してください。)*Map()*の働きは、図 79 に図解されている *Pick()*と類似しています。

*Map()*が呼び出されるためには、下記のようにしてイベントに割り当ててください。

```
*.XtXrtGraph.translations: #override \n\
  ~Shift ~Ctrl <Btn1Down>: Map()
```

*Map()*は、*XrtMap()*の呼び出しで、座標を指定する一つのオプションのパラメータを取ります。デフォルトでは、*Map()*は一番目の Y 軸で指定される座標を使用します。

*XrtMap()*の結果を受け取るには、XtNxrtMapCallback を使用してコールバックを登録してください。各コールバックには、次の構造体のポインタが渡されます。

```
typedef struct {
  int              reason;      /* Read-only */
  XEvent           *event;      /* Read-only */
  String           *params;     /* Read-only */
  Cardinal     nparams;     /* Read-only */
  XrtRegion        region;      /* Read-only */
  XrtMapResult map;         /* Read-only */
} XrtMapCallbackStruct;
```

この構造体のフィールドの変更は意味を持ちません。

例

図 80 は、XRT/graph のデフォルトのズームのトランスレーションを置き換えるアクション・ルーチンの例です。軸の範囲のコントロールに Map コールバックを使用しています。

```
int x1, y1, x2, y2;

static void
start_zoom(Widget w, XEvent *event)
{
    XButtonEvent *be = (XButtonEvent *) event;
    x1 = be->x;
    y1 = be->y;
}

static void
end_zoom(Widget w, XEvent *event)
{
    XrtMapResult min, max;
    int top, bottom, left, right;
    XButtonEvent *be = (XButtonEvent *) event;
    x2 = be->x;
    y2 = be->y;
    left = MIN(x1, x2)
    right = MAX(x1, x2);
    top = MIN(y1, y2);
    bottom = MAX(y1, y2);
    XrtMap(w, 1, left, bottom, &min);
    XrtMap(w, 1, right, top, &max);
    XtVaSetValues(w,
        XtNxrtXAxisMin,XrtFloatToArgVal(min.x),
        XtNxrtYAxisMin,XrtFloatToArgVal(min.y),
        XtNxrtXAxisMax,XrtFloatToArgVal(max.x),
        XtNxrtYAxisMax,XrtFloatToArgVal(max.y),
        0);
}

static char translations[] = "#override \n\
    Ctrl<Btn1Down>: ModifyStart(Zoom) start_zoom() \n\
    Ctrl<Btn1Motion>: ZoomUpdate() \n\
    Ctrl<Btn1Up>: end_zoom() ModifyEnd(Zoom) ";
```

図80　デフォルトのZoomを置き換えるアクション・ルーチン

6.6.2　イベント・ハンドリングの手続き

Xt Intrinsics は、マウスの動きやクリック、キーボードの入力等のイベント・ハンドリングを登録するメカニズムを提供しています。XRT/graph のプログラムは、このメカニズムを利用して、グラフ・ウィジェットに起こるイベントに対応することができます。

次のコードは、Xt Intrinsics で、*my_event_proc()*を登録しています。グラフ・ウィンドウ上でマウス・イベントが起こる度に、*my_event_proc()*が呼び出されます。

```
Widget my_graph;
XtAddEventHandler( my_graph,
  ButtonPressMask | ButtonReleaseMask,
  FALSE, my_event_proc, NULL);
```

イベント手続きは、次のフォーマットを持ちます。

```
void
my_event_proc(
  Widget       w,
  caddr_t      client_data,
  XEvent       *event
  )
{
  /* Deal with the event */
}
```

X イベントのリクエストとハンドリングについては、Xt Intrinsics や Motif 関連の資料を参照してください。

XEvent 構造体には、イベントの pixel 座標があります。XRT/graph は、イベントの pixel 座標を次の個所へマップする手続きを提供しています。

- イベント座標に一番近く表示されているデータのセットとポイントのインデックス
- グラフ座標

XrtPick()

*XrtPick()*は、ユーザがグラフをマウスでクリックした時に"ユーザが何を指しているか"を伝えるメカニズムを提供します。

*XrtPick()*は、グラフのハンドル、データ・セットの指定、(x, y)pixel 座標、`XrtPickResult` 構造体のポインタ、軸のフォーカスを引数として取ります。この手続きは、指定された pixel 座標に一番近いデータ値に関する情報で `XrtPickResult` 構造体をセットします。

```
XrtRegion
XrtPick(
  Widget           graph,
  XrtDsGroup       ds_group,
  int              pix_x,
  int              pix_y,
  XrtPickResult    *pick,
  XrtFocus     focus
)
```

ds_group は、pick の結果が対応するデータ・セット。複合グラフの場合、一つのデータ・セットに pick の結果を制限する時は、`XRT_DATASET1` または `XRT_DATASET2` にセットします。`XRT_DATASET1`｜`XRT_DATASET2` に設定してあれば、どちらかのデータ・セットから一番近いポイントが返されます。複合グラフ以外の場合は、`XRT_DATASET1` にセットしてください。

focus は、XRT_XFOCUS または XRT_YFOCUS にセットし、一番近いポイントを検索する対象を X 軸または Y 軸のどちらかの方向に設定します。X、Y スペースを検索する時は、*focus* を XRT_XFOCUS | XRT_YFOCUS にセットしてください。

*XrtPick()*は、以下の表にある値の内一つを返します。

XRT_RGN_NOWHERE	与えられた pixel 座標は、ピック(pick)できるどの箇所にも近くない。この場合、XrtPickResult はセットされない。
XRT_RGN_IN_GRAPH	与えられた pixel 座標は、グラフに表示されているデータ値の一つに近い。この場合、XrtPickResult 構造体の set と point の値が一番近いデータを示す。distance は、ピックされたデータがどれだけ pixel 座標に近いかを示す。
XRT_RGN_IN_LEGEND	与えられた pixel 座標は、凡例の要素に近い。この場合、set は、どのセットが一番近いかを示す。(point は XRT_RGN_IN_LEGEND) distance は、ピックされたデータが pixel 座標に近いかを示す。
XRT_RGN_IN_HEADER	与えられた pixel 座標はグラフ領域でなくヘッダー領域上にある。ヘッダーがデータ領域に重なっていない限り、XrtPickResult はセットされない。
XRT_RGN_IN_FOOTER	与えられた pixel 座標はグラフ領域でなくフッター領域上にある。フッターがデータ領域に重なっていない限り、XrtPickResult はセットされない。

XrtPickResult 構造体は、次のように定義されます。

```
typedef struct {
  int     pix_x, pix_y;
  int     dataset;
  int     set, point;
  int     distance;
} XrtPickResult;
```

フィールドは次の通りです。

pix_x	*XrtPick()*に渡された x pixel 座標。
pix_y	*XrtPick()*に渡された y pixel 座標。
dataset	結果がどちらのデータセットへ対応するかを示す XRT_DATASET1 か XRT_DATASET2。
set	pixel 座標に一番近く表示されているデータのセット・インデックス。データが表示されていなければ、XRT_RGN_NOWHERE。

point	pixel 座標に一番近く表示されているデータのポイント・インデックス。データが表示されていなければ、XRT_RGN_NOWHERE。
distance	与えられた pixel 座標と、set と point でインデックスされたデータの画面上の距離(pixel)。

ピックされた要素が、円グラフの"その他"のスライスに対応する時は、set は XRT_OTHER_SLICE、point は選択されたポイントを示します。図 81 にあるイベント手続きの例は、クリックされたポイントに一番近い棒を決定し、その値をグラフのヘッダーに表示します。

XrtUnpick()

*XrtUnpick()*は *XrtPick()*と反対の手続きで、データのポイントとセットにあたる pixel 座標を決定します。Xlib ルーチンを利用してグラフ上に描画する場合に使用できます。詳しくは、Appendix D をご覧ください。

```
static void
Select(Widget graph, XEvent *event)
{
    int      idx;
    char     *nhs[2], buffer[100];
    XButtonEvent *button_event;

    XrtPickResultp;
    int rc;
    XrtDataHandletmp_data;

    button_event = (XButtonEvent *) event;

    rc = XrtPick(graph, XRT_DATASET1,
         button_event->x, button_event->y,
         &p, XRT_XFOCUS | XRT_YFOCUS);
    if (rc == XRT_RGN_IN_GRAPH) {
         XtVaGetValues(graph,
              XtNxrtData, &tmp_data, XmNuserData, &idx, NULL);
         sprintf(buffer, "%s -- Day %s: $%.2f",
              header_strings[idx][0],
              tdays_0990[p.point],
              XrtDataGetYElement(tmp_data, p.set, p.point) );
         nhs[0] = buffer;
         nhs[1] = NULL;
         XtVaSetValues(graph, XtNxrtHeaderStrings, nhs, NULL);
    }
}
```

図 81　イベント手続きで XrtPick()を使用する

XrtMap()

*XrtMap()*は、pixel 座標からグラフ座標にマップ(map)します。これはグラフ・ハンドル、軸の指定、(x,y)pixel 座標、マップ構造体へのポインタを取り、pixel 座標、軸の指定、グラフ座標の浮動小数点値をマップ構造体に書き込みます。

yaxis を 1 にセットして *XrtMap()*を呼び出すと、一番目の Y 軸で定義されたグラフ座標にマップします。yaxis を 2 に設定すると二番目の Y 軸で定義されたグラフ座標にマップします。

```
XrtRegion
XrtMap(
  Widget          graph,
  int             yaxis,
  int             pix_x,
  int             pix_y,
  XrtMapResult *map
)
```

XrtMap() は、XRT_RGN_NOWHERE、XRT_RGN_IN_FOOTER、XRT_RGN_IN_HEADER、XRT_RGN_IN_GRAPH を返します。

XrtMapResult 構造体は、次のように定義されます。

```
typedef struct {
  int pix_x, pix_y;
  int yaxis;
  float x, y;
} XrtMapResult;
```

フィールドは次の通りです。

pix_x	*XrtMap()*へ渡された x 座標。
pix_y	*XrtMap()*へ渡された y 座標。
yaxis	*XrtMap()*へ渡された軸の指定。
x	マップされたグラフ座標の X 値。円グラフの場合は、x は XRT_HUGE_VAL。
y	マップされたグラフ座標の Y 値。円グラフの場合は、y は XRT_HUGE_VAL。

図 82 にあるイベント手続きは、マウス左ボタンの down と up のイベントのグラフ座標を取得し、次に指定された矩形の領域のグラフをズームします。ここでは、*fmin()*と *fmax()* が存在すると仮定し、これらは、引数の最小浮動小数点値か最大浮動小数点値を返します。

```
void
zoom_plot(graph, client_data, event)
Widget graph;
caddr_t client_data;
XEvent *event;
{
    XButtonEvent*button_event;
    static float x1, y1;
    XrtMapResult m;
    int rc;

    if (event->type != ButtonPress && event->type != ButtonRelease){
        return();
    }
    button_event = (XbuttonEvent *) event;

    if ( button_event->button == Button1 &&
        XrtMap(graph, 1, button_event->x,
            button_event->y, &m) == XRT_RGN_IN_GRAPH)

        if ( event->type == ButtonPress ) {
            x1 = m.x;
            y1 = m.y;
        }
        else { /* Is a button release */
            XtVaSetValues(my_plot,
                XtNxrtXAxisMin,XrtFloatToArgVal( fmin(x1, m.x) ),
                XtNxrtXAxisMax,XrtFloatToArgVal( fmax(x1, m.x) ),
                XtNxrtYAxisMin,XrtFloatToArgVal( fmin(y1, m.y) ),
                XtNxrtYAxisMax,XrtFloatToArgVal( fmax(y1, m.y) ),
                NULL);
        }
}
```

図82　ズームのイベント手続き

XrtUnmap()

*XrtUnmap()*は*XrtMap()*の反対の手続きで、グラフ座標から pixel 座標にマップします。Xlib ルーチンを利用してグラフ上に描画する場合に使用できます。詳しくは、Appendix D をご覧ください。

6.7 ウィンドウのリサイズ

アプリケーションの多くは、グラフ・ウィジェットを含むウィンドウのリサイズをユーザに許可し、グラフ・ウィジェットを新しいウィンドウのサイズに調節することが必要です。

グラフ・ウィジェットは、Motif Composite Widgets の内部に位置することができます。通常、Composite Widget がグラフ・ウィジェットの最適なサイズを決定し、グラフ・ウィジェットはそれに従って自動的に調節されます。

エクスポースとリサイズのコールバック

アプリケーションは、XtNxrtExposeCallback、XtNxrtResizeCallback を使用してコールバックを指定することにより、グラフ・ウィジェットがいつエクスポーズまたはリサイズされたかを知ることができます。コールバックは、ウィジェットのハンドルとクライアント・データのポインタと共に呼び出されます。三番目のコールバックの引数、call_data は、コールバックの追加の情報を持つ XrtCallbackStruct を指します。詳細は Appendix A を参照してください。リサイズのコールバックは、グラフがリサイズされた後に呼び出され、エクスポーズのコールバックは、グラフが再描画された後に呼び出されます。

アプリケーションは、Xlib の機能を用いてグラフ上に描画する際に、これらのコールバックを使用できます。また、グラフ・ウィジェットのサイズに応じてウィジェットのリソースを調節する時にも、これらのコールバックが使用できます。

次のコードはリサイズのコールバックを記録します。

```
XtAddCallback(mygraph, XtNxrtResizeCallback,
    resize_proc, &my_data);
```

次のリサイズのコールバックは、グラフ・ウィジェットが小さくなりすぎた時に凡例を非表示にします。

```
void
resize_proc(w, client_data, call_data)
Widget        w;
caddr_t       client_data;
XrtCallbackStruct *call_data;
{
  Dimension width;

  XtVaGetValues(w,
    XtNwidth, &width,
    NULL);

  if ( width <= 300 )
    XtVaSetValues(w,
      XtNxrtLegendShow, False,
      NULL);

  else
    XtVaSetValues(w,
      XtNxrtLegendShow, True,
      NULL);
}
```

7

高度なプログラミング

二番目のY軸 ■ 複合グラフ

テキスト・オブジェクト ■ リソース・アップデートのバッチ処理

国際化サポート ■ 高速アップデート機能

MS Windowsへのポーティング ■ 警告メッセージのカスタマイズ

C++の使用 ■ UILの使用

リソース値をストリングに変換する

7.1 二番目のY軸

XRT/graphでは、グラフの左側のY軸(一番目または最初のY軸)に対して、二番目のY軸をグラフの右側に付け加えることができます。グラフが反転(invert)されると、一番目のY軸は下に、二番目のY軸は上に来ます。X軸が逆方向であれば、Y軸の位置は入れ替わります。

二番目のY軸を作成するには二つの方法があります。一つは一番目と二番目のY軸の関係を指定する方法です。もう一つは二番目のデータ・セットを加えて複合グラフを作成する方法です。尚、複合グラフについては7.2で解説します。

グラフに二番目のデータ・セットしか存在しない時は、Y2軸が一番目の軸として扱われ、そのリソースが使用されます。

Y軸の関係を指定する

Y軸の関係は`XtNxrtYAxisMult`と`XtNxrtYAxisConst`のリソースで指定します。XRT/graphは一番目のY軸から次の式を使って二番目のY軸を作成します。

```
Y2 =(Y1 *XtNxrtYAxisMult)+ XtNxrtYAxisConst
```

例えば、摂氏で測定される温度をグラフ化する場合、次のコードで華氏を表す二番目の Y 軸が作成できます。

```
XtVaSetValues(graph,
  XtNxrtYAxisMult,
    XrtFloatToArgVal(9.0/5.0),
  XtNxrtYAxisConst, XrtFloatToArgVal(32.0),
  XtNxrtY2Precision,0,
  XtNxrtY2Title, “Fahenheit",
  NULL);
```

二番目の軸を取り去るには、`XtNxrtYAxisMult` を 0.0 に設定して関係を無効にしてください。

`XtNxrtY2AxisLogarithmic` は、Y 軸の関係が指定されると無視されます。

精度

`XtNxrtY2PrecisionUseDefault` が `True` の時は、XRT/graph が Y2 軸の精度をコントロールします。独自に設定する場合は、二番目の Y 軸の精度を細かく指定しすぎないよう、桁の設定には注意してください。(細かすぎると、注釈が正しく表示されないことがあります。) 通常、Y2 の桁数は次と同じかそれ以上にします。

- Y1 の桁数と `XtNxrtYAxisMult` の示した精度の合計
- `XtNxrtYAxisConst` の示した精度

IEEE 浮動小数点は 6 桁の精度なので、どちらの Y 軸も 6 桁以上の精度を指定しないでください。

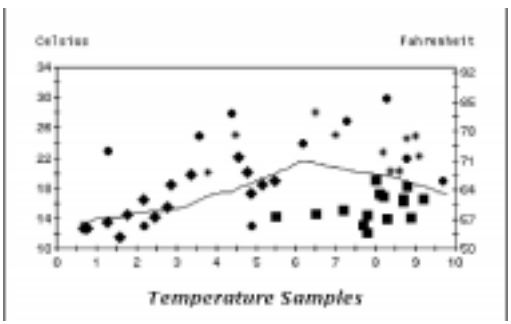


図 83　一番目と linear の関係を持つ二番目の Y 軸

7.2　複合グラフ

複合グラフは、一つのグラフ領域に二つのデータセットを表示します。線、面、棒、積み上げ棒、HiLo、HiLoOpenClose、バブル・チャート、キャンドル・グラフは全て複合グラフに使用できます。

複合グラフの二番目のデータ・セットは、XtNxrtData2 リソースを用いて付け加えられます。グラフのタイプは XtNxrtType2 で設定します(デフォルトは XRT_TYPE_BAR)。デフォルトでは、一番目のグラフが前方に表示されますが、XtNxrtFrontDataset を XRT_DATASET2 にセットすることにより、これを切り替えることができます。マーカも複合グラフに使用することができ、XtNxrtMarkerDataset を使用して、どちらのデータ・セットをマーカの位置決めに使用するか指定できます。

特別なルール

次のルールは、複合グラフにのみ適用されます。

1. どちらかのグラフが、円、ポーラー、レーダー、filled レーダーであれば、二番目のグラフは表示されない
2. X 軸の注釈法がポイント・ラベルであれば、一番目のグラフの(XtNxrtPointLables で指される)ポイント・ラベルのみが使用される

二番目の Y 軸

二番目のグラフの最小、最大、目盛/数値ラベルの増分と精度は、Y2 軸のリソースを使用してプログラミングできます。軸の関係が指定されていなければ、二番目の Y 軸は対数にできます。

二番目のデータ・スタイル

二番目のグラフのデータ・スタイルは、XtNxrtDataStyles2 と XtNxrtDataStyles2UseDefault を使用してプログラミングできます。

凡例

複合グラフの凡例は、各グラフのセット・ラベルが結合して作成されます。二番目のグラフのポイント・ラベルとセット・ラベルは XtNxrtPointLabels2 と XtNxrtSetLabels2 を用いてプログラミングできます。NULL または 0 の長さのストリングのラベルは、凡例には含まれません。

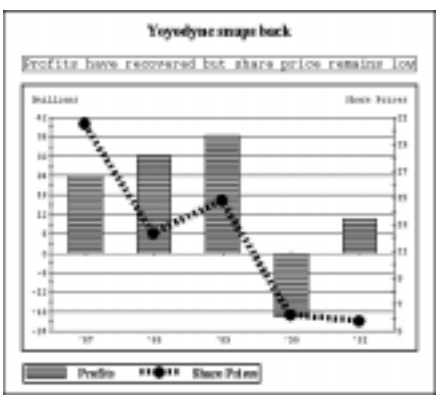


図84　線と棒の複合グラフ

7.3 テキスト･オブジェクト

テキスト･オブジェクトは、グラフ上に描画される、独立した矩形の領域です。このテキスト･オブジェクトは、どのタイプのグラフにも表示することができます。テキスト･オブジェクトは、次の四つの方法でグラフに貼り付けられます。

- 画面上の pixel 位置
- グラフ･データ(X 値と Y 値)
- データ･ポイント(セット、ポイント･インデックス)
- データ･ポイントの上か下(セット、ポイント･インデックス、Y 値)

アプリケーションは、テキスト･オブジェクトを使用して、特定なポイントをハイライトしたり、複数のラベルを表示したりすることができます。テキスト･オブジェクトは、幾つでもグラフに貼り付けることができ、ダイナミックに作成、アップデート、破棄することができます。

テキスト･オブジェクトは、`XrtTextHandle` 型のハンドルで参照されます。XRT/graph には、テキスト･オブジェクトの作成、属性の設定/取得、破棄を行う方法が幾つか提供されています。

$XRTHOME/src/graph/demos/text_area の *text_area.c* にサンプルのコードがありますので参照してください。

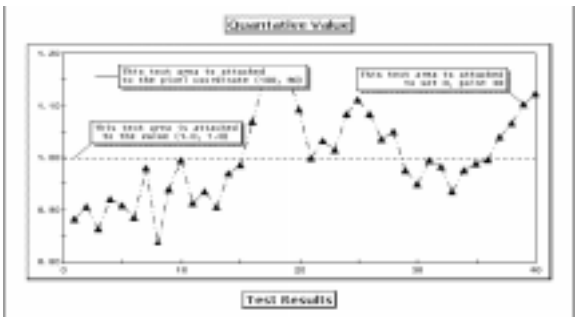


図85　pixel、XY 値、データ･ポイントに貼付けられたテキスト･オブジェクト

テキスト･オブジェクトの作成

グラフにテキスト･オブジェクトを表示するには、次の例のようにオブジェクトをグラフの子として作成してください。

```
XrtTextHandle text_obj;
char *strings[] = {"Hello World!", NULL};

text_obj = XrtTextAreaCreate(graph);
XrtTextAreaSetValues(text_obj, XRT_TEXT_STRINGS, strings, NULL);
```

属性の設定と取得

各テキスト･オブジェクトには、位置と表示をコントロールする属性があります。*XrtTextAreaSetValues()*は、*XtVaSetValues()*と同様に、属性の値をセットします。

また、*XrtTextAreaGetValues()*は、*XtVaGetValues()*と同様に、属性の現在値を取得します。

上記のコードでは、*XrtTextAreaSetValues()*を使用して、テキスト・オブジェクトのテキストをセットしています。図86は、テキスト・オブジェクトの属性の一覧です。これらは、Appendix Bにその詳細が解説されています。

Positioning Attributes

XRT_TEXT_ANCHOR	XRT_TEXT_ATTACH_DATASET	XRT_TEXT_ATTACH_PIXEL_[XY]
XRT_TEXT_ATTACH_POINT	XRT_TEXT_ATTACH_SET	XRT_TEXT_ATTACH_TYPE
XRT_TEXT_ATTACH_VALUE_[XY]	XRT_TEXT_OFFSET	XRT_TEXT_HEIGHT
XRT_TEXT_WIDTH	XRT_TEXT_X	XRT_TEXT_Y

Appearance Attributes

XRT_TEXT_ADJUST	XRT_TEXT_BACKGROUND_COLOR	XRT_TEXT_BORDER
XRT_TEXT_BORDER_WIDTH	XRT_TEXT_FONT	XRT_TEXT_FOREGROUND_COLOR
XRT_TEXT_IS_CONNECTED	XRT_TEXT_IS_SHOWING	XRT_TEXT_ROTATION
XRT_TEXT_STRINGS		

Miscellaneous Attributes

XRT_TEXT_NAME	XRT_TEXT_PRINT_FONT

図86　テキスト・オブジェクトの属性

テキスト

テキスト・オブジェクトに表示されるテキストは、XRT_TEXT_STRINGS で指定されます。テキスト・オブジェクトには、複数行のテキストを表示することができます。

複数行のテキストの行揃えは、XRT_TEXT_ADJUST で指定します。XRT_ADJUST_LEFT、XRT_ADJUST_CENTER、XRT_ADJUST_RIGHT によって、それぞれ左揃え、中央揃え、右揃えに設定できます。

貼り付けるポイントと位置

XRT_TEXT_ATTACH_TYPE を使用して、どのようにテキスト・オブジェクトをグラフに貼り付けるか指定します。オブジェクトは、pixel 座標(XRT_TEXT_ATTACH_PIXEL)、XY 値(XRT_TEXT_ATTACH_VALUE)、データ・ポイント(XRT_TEXT_ATTACH_DATA)、データ・ポイントの上または下の Y 値(XRT_TEXT_ATTACH_DATA_VALUE)に貼り付けることができます。どの方法が選択されたかにより、実際に貼り付けられるポイントをコントロールする属性が異なります。

貼り付け方法	貼り付け位置を指定する属性
pixel 座標	XRT_TEXT_ATTACH_PIXEL_X XRT_TEXT_ATTACH_PIXEL_Y
データ(XY 値)	XRT_TEXT_ATTACH_DATASET XRT_TEXT_ATTACH_VALUE_X XRT_TEXT_ATTACH_VALUE_Y
データ・ポイント	XRT_TEXT_ATTACH_DATASET XRT_TEXT_ATTACH_POINT XRT_TEXT_ATTACH_SET
データ・ポイントの上または下の Y 値	XRT_TEXT_ATTACH_DATASET XRT_TEXT_ATTACH_POINT XRT_TEXT_ATTACH_SET XRT_TEXT_ATTACH_VALUE_Y

テキスト・オブジェクトの貼り付けポイントが決定したら、XRT_TEXT_ANCHOR と XRT_TEXT_OFFSET を使用して、位置を設定することができます。テキスト・オブジェクトは、コンパス(方位磁石)の八つの方向、ポイントの中心または、XRT/graph が選ぶ最適な位置に配置させることができます。

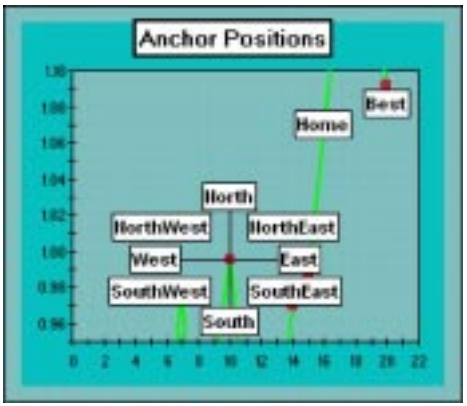


図87　アンカーを使用した位置決め

次の例は、テキスト・オブジェクトを指定位置に貼り付けしています。

```
XrtTextAreaSetValues(my_text,
  XRT_TEXT_ATTACH_TYPE, XRT_TEXT_ATTACH_DATA,
  XRT_TEXT_ATTACH_POINT,    3,
  XRT_TEXT_ATTACH_SET,      2,
  XRT_TEXT_ANCHOR,      XRT_ANCHOR_NORTH,
  XRT_TEXT_OFFSET,      10,
  NULL);
```

接続線

テキスト・オブジェクトとその貼り付けポイントは、XRT_TEXT_IS_CONNECTED を使用して線で結ぶことができます。この属性は、XRT_TEXT_OFFSET と使用して、テキスト・オブジェクトと貼り付けポイントの距離を設定してください。

フォント

テキスト・オブジェクトが作成される時は、XtNxrtAxisFont と同じフォントを使用します。このフォントを変更する場合は、XRT_TEXT_FONT を使用してください。この属性は、ヘッダー、フッター、凡例のフォントと同じ値に指定できます。尚、詳細は、4.8 を参照してください

枠線

テキスト・オブジェクトには、XRT_TEXT_BORDER と XRT_TEXT_BORDER_WIDTH を使用して、枠線を表示することができます。XRT/graph で使用できる枠線であれば、どれでも使用できます。(枠線は、4.16 に解説されています。)

色

XRT_TEXT_BACKGROUND_COLOR と XRT_TEXT_FOREGROUND_COLOR を使用して、テキスト・オブジェクトの背景色と前景色をセットできます。(色の指定は、2.8 に解説されています。)

イメージ

XRT_TEXT_IMAGE、XRT_TEXT_IMAGE_LAYOUT、XRT_TEXT_IMAGE_MINIMUM_SIZE を使用して、テキスト・オブジェクトの背景にイメージをセットすることができます。尚、詳細は、P91 を参照してください。

テキストの回転

テキスト・オブジェクトのテキストは、XRT_TEXT_ROTATION を使用して、回転させることができます。90 度(XRT_ROTATE_90)、270 度(XRT_ROTATE_270)または回転なし(XRT_ROTATE_NONE)にセットできます。

テキスト・オブジェクトのパフォーマンス

アプリケーションでは、複数のテキスト・オブジェクトの作成や操作は、バッチ処理で行ってください。尚、バッチ処理は、下記の 7.4 に解説されています。

テキスト・オブジェクトの削除

テキスト・オブジェクトを画面から非表示にするには、XRT_TEXT_IS_SHOWING を False にセットしてください。オブジェクトを破棄するには、*XrtTextAreaDestroy()*を呼び出します。

7.4 リソース・アップデートのバッチ処理

通常、リソース変更のアップデートは、*XtSetValues()*や *XtVaSetValues()*が呼び出された直後に行われます。グラフが再描画される前に複数のリソースに変更を加える場合は、XtNxrtRepaint を False にセットします。XtNxrtRepaint が True にセットされるまで、全てのリソースの変更は一括してまとめられます。

複数のテキスト・オブジェクトの作成やアップデートは、一括処理した方が良い結果が得られます。

7.5 国際化サポート

Locale

XRT/graph は、ANSI と POSIX の *locale* 標準をサポートしています。特に、時間軸のフォーマットには *LC_TIME* 環境変数が使用されます。日本語が表示されるようにするには、日本語の *locale* で実行されている必要があります。また、プログラムの最初に *XtSetLanguageProc()* を呼び出して、アプリケーションが必ず *locale* をイニシャライズするようにしてください。尚、*setlocale()*や *XtSetLanguageProc()*の設定や初期化については、X Window 関連資料を参照してください。

フォント

X サーバでサポートされている Motif のフォント・リストが、グラフのヘッダー、フッター、凡例、注釈、テキスト領域に使用できます。日本語の文字は、XmFontList に必要な日本語フォントが含まれていれば、正しく表示できます。

ISO Latin 1 は欧米の言語の標準文字セットです。X Window フォントの内、名前の最後に *iso8859-1* とつくものは、ISO Latin 1 フォントです。プログラムが"C"の *locale* で実行されている時は、一つの ISO Latin 1 を持つフォント・リストだけで充分です。

XRT/graph の印刷手続き *XrtDrawPS()*は、日本語テキストも PostScript に出力することができます。

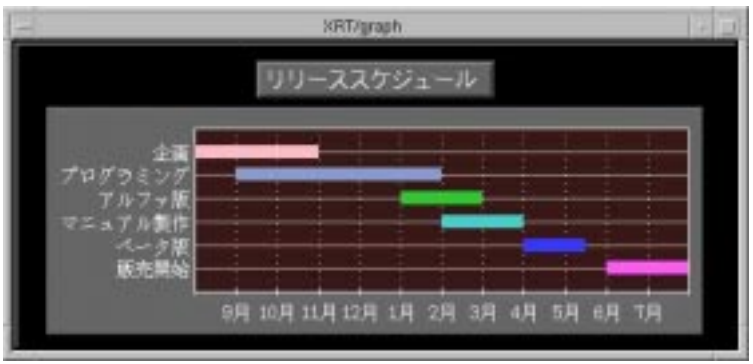


図 88　LANG が"japanese"にセットされているグラフ

7.6 高速アップデート機能

通常、グラフに新しいデータを加える場合は、グラフのデータ・オブジェクトをアップデートし、XtNxrtData と(または)XtNxrtData2 のリソースを該当する XrtDataHandle へリセットします。その時、XRT/graph はデータ全体を調べ、グラフ全体を再描画します。

高速アップデート機能は、既存のグラフに効率よく新しいデータを加える方法です。この機能は、新しいデータがグラフの現在の軸の範囲にある場合にのみ効果があります。もし新しいデータが軸の範囲外であれば、グラフ全体は再描画されます。[1]

以下は、高速アップデートの手続きの一覧です。これらは、Appendix D に詳細が解説されています

XrtArrDataFastUpdate()	Array(配列)データ・セットの各セットに加えられた新しいポイントを表示する。
XrtGenDataFastUpdate()	General(一般)データ・セットの一つのセットに加えられた新しいポイントを表示する。
XrtArrCheckAxisBounds()	Array データ・セットの各セットに加えられた新しいポイントをチェックし、そのポイントがグラフ化されたらどの軸の範囲を越えるかを調べる。
XrtGenCheckAxisBounds()	General データ・セットの一つのセットに加えられた新しいポイントをチェックし、そのポイントがグラフ化されたらどの軸の範囲を越えるかを調べる。
XrtArrDataShiftPts()	(全てのセットの)既存のポイントを左側にコピーする。軸の範囲が越えられた時に使用。Shift-out された(移動によって空白になった)ポイントは、hole 値にセットされる。
XrtGenDataShiftPts()	(一つのセットの)既存のポイントのコピーを左側に作成する。軸の範囲が越えられた時に使用。Shift-out された(移動によって空白になった)ポイントは、hole 値にセットされる。

高速アップデート機能を使用するには、次の要領に従ってください。

- XtNxrtData または XtNxrtData2 で参照されている、既存のデータ・オブジェクトに新しいポイントを追加する。
- アップデートが高速に実行されるかを事前に調べる場合は、該当する CheckAxisBounds 手続きを呼び出す。
- 該当する FastUpdate 手続きを呼び出してアップデートを実行する。

[1] 高速アップデート手続きは、新しいデータが軸の範囲外に来た場合にのみ expose コールバックを開始します。

切れ目のあるパターン(点線や破線)は、高速アップデートを使用した場合、正しく表示されない場合があります。この機能が必要なアプリケーションには切れ目のない線を使用してください。

次のコードは、高速アップデート機能を使用して scrolling strip chart がどのように実行されるかを示す例です。

```
1 if (XrtArrDataAppendPts(data, new_xvalue, new_yvector)) {
2   bound = XrtArrCheckAxisBounds(graph, 1, 1); /* new points fit in axes? */
3        if ((bound & XRT_GTX) == 0) { /* less than xmax? */
4             XrtArrDataFastUpdate(graph, 1, 1); /* Fast Update */
5        }
6        else {
7             new_numpoints = XrtDataGetNPoints(data, 0) - increment;
8             /* shift X values */
9             XrtArrDataShiftPts(data, 0, increment, new_numpoints);
10            XrtDataSetNPoints(data, 0, new_numpoints);
11            xmin = XrtDataGetXElement(data, 0, 0);
12            xmax = XrtDataGetXElement(data, 0, new_numpoints-1) +
13                 increment * x_increment;
14            XtVaSetValues(graph, /* repaint with new axis */
15                 XtNxrtData, data,
16                 XtNxrtXMin, XrtFloatToArgVal(xmin),
17                 XtNxrtXMax, XrtFloatToArgVal(xmax),
18                 NULL);
19       }
20 }
```

図 89　Scrolling Strip Chart のコード

1 行目で、新しいデータ・ポイントがグラフのデータ・オブジェクト(このハンドルは、グラフの `XtNxrtData` の現在値)に加えられます。

2 行目は、新しいポイントが現在の軸の範囲に収まるかを調べます。収まる場合は、高速アップデート機能が呼び出されます(4 行目)。収まらない場合は、データセット全てが下にシフトされ(7-13 行目)、そしてグラフ化(14-18 行目)されます。

7.7　MS Windows へのポーティング

KL Group 社の Olectra Chart という製品を使用して、XRT/graph のアプリケーションは、Microsoft Windows にポーティングすることができます。Olectra Chart を使用することにより、グラフ・アプリケーションは、ネイティブの Windows ツールキットを使用し、Windows 3.1、Windows 95、Windows NT に対応することができます。このセクションでは、XRT/graph のクロス・プラットフォーム API や、二つのプラットフォームの違いの処理の方法等を解説します。

概要

Windows ツールキットには、Motif と同様、定義済みのコントロールがあります。Olectra Chart の 16 と 32 bit の"2D Chart"DLL カスタム・コントロールは、グラフを作成する"ウィジェット"(コントロール)を Windows ツールキットに追加します。

このコントロールは、XRT/graph と同じリソース(Windows の場合はプロパティ)、データ型、方法を持ちます。[1] 共通の API を使用することにより、XRT/graph アプリケーションをリコンパイルし、短時間で Windows にポーティングさせることができます。アプリケーションは、Motif または Windows 特有のコードを処理する場合、`#ifdef` ステートメントを使用することができます。

C++ラッパーを使用すると、アプリケーション・コードのプラットフォームへの依存が少なくなります。XRT/graph と Olectra Chart には、どちらも、C++ラッパーが提供されています。

グラフ・コントロールの作成

Windows のプログラムでは、コントロールは異なる方法で作成されます。Motif ウィジェットは、他のウィジェットの子として作成(instantiate)され、サイズや位置は作成時に定義されますが、Windows では、通常、コントロールは他のコントロールとは独立して、リソースまたは*RC* ファイルで定義されます。

2D Chart コントロールを定義する時に使用されるクラス名は、"OlectraChart2D"(32-bit DLL)または"OlectraChart2D16"(16-bit DLL)です。

コントロールが一度定義された後は、Windows は、プログラムの初期化中にコントロールを作成することができます。(XRT/graph を toplevel シェルの子として作成する)*XtCreateManagedWidget()*の代りに、Olectra Chart のプログラムは次のようなコードを使用します。

```
static HXRT2D graph;
static HWND hwndChart;

hwndChart = GetDlgItem(hWnd, IDCHART); /* defined in .RC file */
graph = XrtCreate();
XrtAttachWindow(graph, hwndChart); /* joins chart and window /*
```

一つの `Widget` ハンドルに対して、Windows プログラムは 2D Chart コントロール(`HXRT2D`)と画面上のウィンドウ(`HWND`)の二つのハンドルが必要です。

グラフ・データ

XRT/graph と Olectra Char 2D Chart では、グラフ・データのデータ型が異なります。XRT/graph は `float` を使用しますが、2D Chart は `double` を使用します。これを処理するには、下の例のように、Motif と Windows 両方のデータ型の定数を定義してください。

```
#ifdef WIN_DLL
#define XrtFloat double
#endif
#ifdef MOTIF
#define XrtFloat float
#endif
```

[1] Olectra Chart には OCX も提供されていますが、DLL のみが XRT/graph とのクロス・プラットフォーム API を共有します。

リソースの設定と取得

XRT/graph と Olectra Chart は、リソース値の設定と取得に共通の API を使用します。*XrtSetValues()*は *XtVaSetValues()*と同様に、一つまたは複数のリソースを設定することができます。*XrtGetValues()* は *XtVaGetValues()* と同様に、リソースの現在値を取得します。*XtVaSetValues()*を呼び出してグラフのリソースをセットする Motif のコードは、*XrtSetValues()* を呼ぶように変更することができます。(注意:`float` 型のリソースにアクセスする前は、*XrtFloatToArgVal()*または *XrtArgValToFloat()*を呼び出さないでください。)

各 XRT/graph のリソースは、Motif と Windows の 2D Chart コードで使用できる、クロス・プラットフォーム用の名前を持っています。図 90 は、幾つかのリソースの例です。全リストの一覧は、P267 にあります。

XRT_AXIS_FONT	XRT_BACKGROUND_COLOR	XRT_DATA	XRT_DATA_STYLES
XRT_FOOTER_BORDER	XRT_FOOTER_STRINGS	XRT_GRAPH_HEIGHT	XRT_HEADER_FONT
XRT_LEGEND_SHOW	XRT_POINT_LABELS	XRT_SET_LABELS	XRT_TYPE
XRT_XANNOTATION_METHOD	XRT_YAXIS_MAX	XRT_YLABELS	XRT_Y2TITLE

図90　クロス・プラットフォームのリソース名

次の例は、(`my_data` と `labels` 変数が正しく定義されていることを前提として)三つのグラフ・プロパティをセットしています。

```
XrtSetValues(graph,
  XRT_DATA,        my_data,
  XRT_TYPE,        XRT_TYPE_STACKING_BAR,
  XRT_SET_LABELS,  labels,
  NULL);
```

ストリングを使用したリソース設定

*XrtSetPropString()*と *XrtGetPropString()*を使用して、アプリケーションのポーティング作業を削減することもできます。これらの手続きは、ストリング・ポインタとしてリソースを設定/取得しますので、例えば、Windows の `COLORREF` を使用するようコードを書き直さずに、色を設定することができます。値のフォーマットは、XRT/graph のリソース・ファイルの場合と同じです。(リソース・ファイルの構文は Appendix H にあります。)次の例では、ヘッダーの背景色をセットしています。

```
XrtSetPropString(graph,
  XRT_HEADER_BACKGROUND_COLOR, "orange");
```

データ型のハンドリング

XRT/graph と Olectra Chart 2D Chart のプログラミングで使用されるデータ型は、ほとんど同じです。但し、二つのプラット・フォームの違いから、多少の相違点があります。XRT/graph と Olectra Chart には、これらのデータ型に対して、次のクロス・プラットフォーム `typedefs` が提供されています。

ポーティング可能な typedef	XRT/graphの データ型	Olectra Chartの データ型
	`Widget`	`HXRT2D`

ポーティング可能な typedef	XRT/graphの データ型	Olectra Chartの データ型
	`float`	`Double`
XrtBoolean	`Boolean`	`BOOL`
XrtColor	`char *`	`COLORREF`
XrtDimension	`Dimension`	`int`
XrtFont	`XmFontList`	`HFONT`
XrtPosition	`Position`	`int`

クロス・プラットフォームの例

XRT/graph には、Motif と Windows の両方でコンパイルできる *common.c* というプログラムがあります。(図91 参照) *common.c* は、*$XRTHOME/src/graph/examples* にあります。このプログラムを Windows でコンパイルするには、次の手順に従ってください。

1 Olectra Chart をインストールし、開発環境にコンフィギュアする。Olectra Chart の *\CHART\2D\DEMOS\DLL\SDK* ディレクトリにあるサンプル・コードをコンパイル、実行して、この作業が正しく行われているか確認できる。

2 *common.c*、*common.rc*、*olch2d.ico*、*resource.h*、*makefile.w32* を Windows にコピーする。16bit の開発環境を使用している場合は、*makefile.w16* と *common.def* もコピーする。(makefile で指定されるディレクトリは、*CHART\2D\COMMON*)

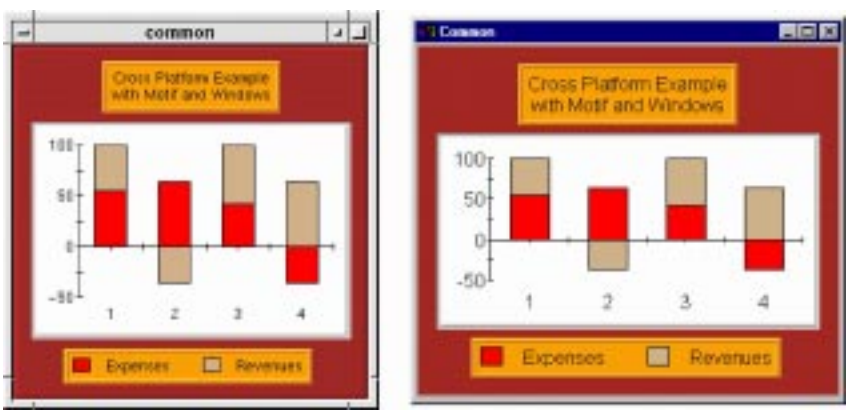


図91 Motif と Windows での common.c

7.8 警告メッセージのカスタマイズ

XRT/graph は、警告メッセージの出力に *XtAppWarningMsg()*を使用します。Xt エラー・データベースでは、警告のテキストをアプリケーションとは独立してカスタマイズすることができます。XRT/graph のメッセージは、*$XRTHOME/include/Xm/XrtGraph.err* にあります。

警告やエラー・メッセージを変更するには、このファイルのコピーを編集し、次のようにしてください。

- 変更したメッセージをシステムのメッセージ・ファイル(*/usr/lib/X11/XtErrorDB*)に追加する。または、
- Xlib ルーチンの *XrmMergeDatabases()*を呼び出して、変更したファイルとシステムのファイルを実行時にマージする。

7.9 C++の使用

XRT/graph には、C++のインターフェースが提供されています。これを使用するには、ヘッダー・ファイル*$XRTHOME/include/Xm/XrtGraph.hxx* を C++プログラムにインクルードしてください。このファイルは、次の三つのクラスを作成します。

クラス	方法
XrtGraph	■ *XtCreateXrtGraph()*を呼ぶ constructor。 ■ *XtDestroyWidget()*を呼ぶ destructor。 ■ XRT/graph を最初の引数として取る各 C ルーチンに対応する方法。(例: *XrtDrawPS()*の方法は *drawPS()*) ■ 各設定可能なリソースをセットする方法。(例: `XtNxrtType` をセットする方法は *setType()*) ■ 各リソースを取得する方法。(例: `XtNxrtBackgroundColor` を取得する方法は *getBackgroundColor()*) ■ 一般的な Xt のファンクションを実行する方法。(*addCallback()*、*destroy()*、*getDisplay()* 、 *getName()* 、 *getParent()* 、 *isManaged()* 、 *isRealized()* 、 *manageChild()*)
XrtGraphData	■ *XrtDataCreate()* 、 *XrtDataCreateFromFile()* 、 *XrtDataCopy()* を呼ぶ constructor。 ■ *XrtDataDestroy()*を呼ぶ destructor。 ■ データ・オブジェクトのハンドルを最初の引数として取る、ほとんどの C ルーチンに対応する方法。(例:*XrtDataSetYElement()*の方法は *setYElement()*) ■ *XrtDataCopy()*を実行する、Overloaded = operator ■ `XrtDataHandle` の Casting サポート
XrtTextArea	■ *XrtTextCreate()*を呼ぶ constructor。 ■ *XrtTextDestroy()*を呼ぶ destructor。 ■ 各設定可能なテキスト・オブジェクトの属性をセットする方法。(例:XRT_TEXT_ATTACH_TYPE をセットする方法は *setAttachType()*) ■ 各テキスト・オブジェクトの属性を取得する方法。(XRT_TEXT_ATTACH_TYPE を取得する方法は *getBackgroundColor()*)

クラス	方法
XrtPointStyle	■ *XrtPointStyleCreate()*を呼ぶ constructor。 ■ *XrtPointStyleDestroy()*を呼ぶ destructor。 ■ 既存の XrtPointStyle をポイントとセットによって取得する constructor。 ■ 各設定可能な PointStyle リソースをセットする方法。(例: XRT_POINTSTYLE_DISPLAY を取得する方法は *setDisplay()*) ■ 各 PointStyle リソースを取得する方法。(例: XRT_POINTSTYLE_SLICE_OFFSET を取得する方法は *getSliceOffset()*)

簡単な C++の例が、*$XRTHOME/src/graph/examples/bar2tran_cpp.cxx* にあります。

7.10 UIL の使用

XRT/graph ウィジェットの作成には、OSF の User Interface Language(UIL)を使用することができます。UIL は、標準の Motif ウィジェットに使用されていないデータ型はサポートしていないので、XRT/graph のリソースの中には UIL でセットできないものもあります。但し、これらは、`MrmNcreateCallback` を用いればアプリケーションでセットすることが可能です。尚、UIL の引数とデータ型の解説は、Appendix I にあります。

UIL でグラフを作成するには、次の手順に従ってください。

- XRT/graph の UIL ヘッダー・ファイル、*XrtGraph.uil* をインクルードする。
- *MrmInitialize()*を呼び出した直後に、*XrtMrmInitialize()*を呼ぶ。

$XRTHOME/src/graph/examples には、UIL のサンプル・プログラム(*simple_uil.c* と *simple_uil.uil*)があります。

7.11 リソース値をストリングに変換する

リソース変換プログラムは、リソース値のフォーマットを変換します。このプログラムは、通常、リソース・ファイルにある値をストリングから、ウィジェットが理解できるフォーマットに変換する場合に使用されます。(例: Boolean と整数) XRT/graph は、装備されたこのプログラムを自動的に登録し、ストリングを XRT/graph 固有のリソース型(`XtRXrtDataStyle` や `XtRXrtAlign` 等)に変換します。

XRT/graph には、XRT/graph 固有のリソース型からストリングに変換する"逆変換プログラム"も提供されています。リソース値をリソース・ファイルに書く時や、ウィジェットの状態を保管する時に便利です。リソース逆変換プログラムは、変換の結果の格納に必要なスペースを割り当てます。このストリングは、使用後、必ず解放してください。

グラフ・ウィジェットが作成される前に、変換プログラムを使用可能に設定するには、*XtInitializeWidgetClass()*を呼び出し、逆変換プログラムだけを使用可能に設定するには、*XrtSetReverseConverters()*を呼び出してください。

次のコードは、凡例の方向の値をプリントする時、これらの変換プログラムがどのように使用できるかを示しています。

```
void
print_legend_orientation()
{
 XrtAlign     align;
 XrmValue     from, to;

 XrtSetReverseConverters();

 XtVaGetValues(graph,
     XtNxrtLegendOrientation, &align, NULL);
 from.addr = (caddr_t) &align;
 from.size = sizeof(align);
 to.addr = NULL;
 if (XtConvertAndStore(graph, XtRXrtAlign, &from,
          XtRString, &to)) {
     printf("Legend Orientation: %s\n", to.addr);
     XtFree(to.addr);
   }
}
```

または、*XrtGetPropString()*を使用することもできます。

```
void
print_legend_orientation()
{
 string  string;
 if (XrtGetPropString(graph, XRT_LEGEND_ORIENTATION, &string))
{
      printf("Legend Orientation: %s\n", string);
      XrtFreePropString(string);
 }
}
```

8

PDS ユーティリテイ・ライブラリ

XrtPDSImage ■ Image 型の登録

ダイナミックまたは共用ライブラリの使用

XrtPDSList オブジェクト ■ 手続きと方法

8.1 XrtPDSImage

XrtPDSImage は、イメージと一緒にファイル名や、幅、高さなどのプロパティを持つオブジェクトです。このオブジェクト内部では、Pixmap フォーマットが使用されています。

XRTPDSImage には、ファイル名や、幅、高さ、clip mask、Pixmap 自身を返すパブリック・ファンクションがありますが、これらの属性はセットすることはできません。

XrtPDSImage を利用することにより、XRT ウィジェットは、イメージ・オブジェクトを処理するだけでよくなり、そのイメージがどこにあるか、元のフォーマットが何であるかといったことを気にする必要がなくなります。また、新しいイメージの型も、ウィジェットのライブラリを変更することなしに追加することができます。

8.1.1 XrtPDSImage の作成

Pixmap から XrtPDSImage を作成するには、*XrtPDSImageCreate()*を呼び出してください。

```
image = XrtPDSImageCreate(pixmap, display, width, height);
```

pixmap は、指定されたサイズに変更されたり切り取られたすることがないので、この場合の *width* と *height* は、pixmap の実際の幅と高さにしてください。

通常、XRTPDSImage は pixmap からユーザによって直接作成されることはありません。イメージはサポートされるフォーマットを使用してファイルに記憶されます。XRT/graph では、ウィジェットに登録されているコンバータを使用して XrtPDSImage が作成されます。デフォルトでは、これらのコンバータは、XPM、XWD、GIF ファイルを XrtPDSImage に変換することができます。PNG と JPEG を使用するには、ウィジェットにこれらのイメージ型を登録し、明示的に使用可能になるよう設定してください。尚、これ以外のイメージ型を使用する場合は、8.2.2 を参照してください。

8.2 Image 型の登録

XrtPDSImage オブジェクトを使用することができる XRT ウィジェットは、登録されたイメージ型のリストを持っています。登録されていない型や新しい型は、次のファンクションを使用して登録できます。

```
Boolean
XrtEnableImageType(
  XtPointer    class,
  XtPointer    functions
  )
```

class はウィジェット・クラス(i.e. `xtXrtGraphWidgetClass`)は、*function* は `XrtPDSImageFuncs` 構造体へのポインタです。この構造体は次のように定義されます。

```
typedef struct {
  XrtPDSImageNameCompareProc    check_name;
  XrtPDSImageHeaderCompareProccheck_header;
  XrtPDSImageCreateProc         create_image;
  XrtPDSImageType               type;
  XrtPDSImageOutputProc         output;
} XrtPDSImageFuncs
```

例えば、JPEG の入出力は、次のようにして登録することができます。

```
#include <Xm/XrtGraph.h>
#include <Xm/XrtPDSJpeg.h>
XrtEnableImageType((XtPointer)xtXrtGraphWidgetClass,
  (XtPointer)XrtPDSJpegImageUtils);
```

XrtPDSJpegImageUtils は、定義済みの `XrtPDSImageFuncs` 構造体のポインタです。この後、*libxrtjpeg.a* をアプリケーションにリンクしてください。

イメージ型が登録されると、この型は、ウイジェットが持つイメージ型のリストに追加されます。ファイルからイメージをロードするように要求が出されると、ウィジェットは、`XrtPDSImageFuncs.check_name` を呼び出してこのリストを検索し、このファイル名の型がサポートされているかを調べます。失敗時には、`XrtPDSImageFuncs.check_header` が呼び出され、ファイルの最初の 250byte からファイルのイメージ型が識別できるかどうか調べます。

ファイルのイメージ型をサポートする `XrtPDSImageFuncs` 構造体が検索されると、`XrtPDSImageFuncs.create_image` が呼び出され、そのイメージ・ファイルから XrtPDSImage が作成されます。

注意: `XrtPDSImageFuncs` 構造体の型は、この構造体でサポートされているイメージの型を表す int です。

8.2.1 定義済みのイメージ

使用できる定義済みのイメージは次の通りです。

```
#define XRTPDS_XPM_IMAGE 0
#define XRTPDS_XWD_IMAGE 1
#define XRTPDS_GIF_IMAGE 2
#define XRTPDS_JPEG_IMAGE 3
#define XRTPDS_PNG_IMAGE 4
```

デフォルトでは、XPM、XWD、GIF ファイルの読み込みと、XPM と XWD ファイルの書き込みが、XRT/graph と XRT/3d でサポートされています。PNG と JPEG の入出力は、該当するライブラリをリンクしイメージ型を登録することによってサポートされるようになります。JPEG ライブラリは *libxrtjpeg.a*、PNG ライブラリは *libxrtpng.a* です(*libz.a* 圧縮ライブラリも必要です)。

イメージのサポートは、イメージ・オブジェクトと `XrtPDSImageFuncs` 構造体の二つで行われます。イメージ・オブジェクトは、変換されたイメージを持ち、`XrtPDSImageFuncs` 構造体は、イメージの読み書きを処理する方法を持ちます。

各イメージ・フォーマットは、`XrtPDSImageFuncs` 構造体を持ちます。(例: XPM は `XrtPDSXpmImageUtils`、JPEG は `XrtPDSJpegImageUtils`) 例えば、JPEG イメージから XrtPDSImage を作成する時は、`XrtPDSJpegImageUtils.create_image` を呼び出すと、変換されたイメージを持つオブジェクトが返されます。

8.2.2 任意のイメージ

任意のイメージ・フォーマットの読み書きは、`XrtPDSImageFuncs` に該当するフィールドを設定することにより、サポートすることができるようになります。下記は、構造体とそのフィールドの例です。

```
typedef struct {
  XrtPDSImageNameCompareProccheck_name;
  XrtPDSImageHeaderCompareProccheck_header;
  XrtPDSImageCreateProc     create_image;
  XrtPDSImageType               type;
  XrtPDSImageOutputProc     output;
} XrtPDSImageFuncs;
```

例えば、Targa(.TGA)ファイルを使用する場合は、下記のように定義を作成し、ファンクションを書く必要があります。

```
/* Use a number greater than the maximum predefined type */
#define TARGA_IMAGE_TYPE 100

Boolean
CheckIsTargaFilename(
 string   filename;
 )

Boolean
CheckTargaMagicNumber(
 char     *buffer
 )

XrtPDSImage
CreateImageFromTargaFile(
 Widget   w,
 Ximage   *image,
 String   filename,
 Boolean  color,
 String   msg
 )

Boolean
WriteImageToTargaFile(
 Widget   w,
 Ximage   *image,
 String   filename,
 Boolean  color,
 String   msg
 )
```

次にXrtPDSImageFuncs 構造体を作成し、セットしてください。

```
XrtPDSImageFuncs TargaImageUtilStruct = {
 (XrtPDSImageNameCompareProc) CheckIsTargaFilename,
 (XrtPDSImageHeaderCompareProc) CheckTargaMagicNumber,
 (XrtPDSImageCreateProc) CreateImageFromTargaFile,
 (XrtPDSImageType) TARGA_IMAGE_TYPE,
 (XrtPDSImage) WriteImageToTargaFile
};
XrtPDSImageFuncs *TargaImageUtils = &TargaImageUtilStruct
```

最後に、この新しいイメージ型を登録します。

```
XrtEnableImageType(xtXrtGraphWidgetClass, TargaImageUtils)
```

これで、(XtNxrtDataAreaImage 等の)リソースの Targa イメージを読み込むことができるようになりました。また、Targa イメージは、*XrtSaveImageGeneric()*を呼び出すことによって書き込むこともできます。

```
XrtSaveImageGeneric(graph, TARGA_IMAGE_TYPE, "output.tga", NULL)
```

8.2.3 イメージ・ファンクションの書き込み

XrtPDSImageNameCompareProc

```
Boolean
CheckIsTargaFilename(
  String       file_name
  )
```

このファンクションには、イメージのファイル名が渡されます。ファイルの名前や拡張子が有効であれば`True`、そうでなければ`False`を返します。

XrtPDSImageHeaderCompareProc

```
Boolean
CheckTargaMagicNumber(
  char *buffer
  )
```

このファンクションには、イメージ・ファイルの最初の 250byte を持つバッファが渡され、このファイルが有効であるかどうかが調べられます。通常は、ファイルの最初の部分にあるマジック・ナンバが調べられます。(例: GIF の場合は"GIF87a"、"GIF89a") ファイルが有効であればTrue、そうでなければFalse が返されます。

XrtPDSImageCreateProc

```
XrtPDSImage
CreateImageFromTargaFile(
  Widget       w,
  String       file_name
  )
```

このファンクションには、ウィジェットとイメージ・ファイルの名前が渡され、このファイルからXrtPDSImage が作成されます。通常、この作業は次のように行われます。

1 イメージをネイティブ・フォーマットでロードする

2 Pixmap に変換する
(Xpm ライブラリにあるファンクションを利用すると便利です。)

3 *XrtPDSImageCreate()*でこの pixmap より XrtPDSImage を作成する

XrtPDSImageOutputProc

```
Boolean
WriteImageToTargaFile(
  Widget   w,
  Ximage   *image,
  String   filename,
  Boolean  color,
  String   msg
  )
```

このファンクションには、ウィジェット、変換され保管される Ximage、イメージが保管されるファイルの名前、白黒かカラーかを示す Boolean、エラー・メッセージ用のメッセージ・バッファが渡されます。バッファが `NULL` であれば、エラー・メッセージは、`stdout` または `stderr` にプリントされます。

8.3 ダイナミックまたは共用ライブラリの使用

JPEG と PNG イメージ・ライブラリは、スタティック(静的)ライブラリとして提供されています。ダイナミック・ライブラリや共用ライブラリを作成する時は、(インターネットから無料でダウンロードできる)ソースコードを取得し、リコンパイルする必要があります。各イメージ・ライブラリには、カスタマイズされた XRT モジュールが追加されていますが、それ以外は変更されていません。このモジュールは、共用ライブラリに簡単に追加できるよう、コンパイルされた形で提供されています。（このモジュールは、*$XRTHOME/lib-shared* にあります。）

このモジュールは、XRT PDS の共用ライブラリが用意されている下記の機種にのみ提供されています。

SGI IRIX 5.x	SGI IRIX 6.x
SGI IRIX 6.x 64-bit	HP 9000 Series 700/800
SPARC SunOS 4.x	Alpha/OSF
SPARC Solaris 2.x	SPARC Solaris 7.x 64-bit
SPARC Solaris 7.x	IBM RS6000 AIX 4.x
IBM RS6000 AIX 4.x 64-bit	

8.4 XrtPDSList オブジェクト

XrtPDSList オブジェクトは、同じサイズのアイテムのグループを記憶させる時に便利なオブジェクトです。提供されている方法を使用して、これらのアイテムの追加、削除、並べ替え、取得等を簡単に行うことができます。

XrtPDSList オブジェクトで指定されるリストは 0 ベースです(最初のエレメントのインデックスは0、二番目のエレメントのインデックスは 1….というように続きます)。

8.4.1 サンプル･コード

下記は、リストを作成し、アイテムを追加、そして取得している例です。

```
void
CreateListOfIntegers()
{
 XrtPDSList list;
 int i, *ptr;

 list = XrtPDSListCreate(sizeof(int));
 i = 34;
 XrtPDSListAppend(list, &i);
 i = 46;
 XrtPDSListAppend(list, &i);
 ptr = XrtPDSListGetItem(list, 1);
 fprintf (stderr, "item is %d\n", *ptr);
}
```

この手続きでは整数のリストを作成し、整数 34 と 46 を追加しています。次にリストの二番目のアイテムを取得しそれを表示しています。

8.4.2 リストの作成と破棄

リストを作成するには、*XrtPDSListCreate()*を呼び出してください。

```
list = XrtPDSListCreate(sizeof(int));
```

この方法には、リストの各エレメントのサイズを表すパラメータが渡されます。(異なる型のエレメントは、サイズが同じであれば同じリストに記憶させることができます。)この方法は、作成されたリストを返します。

*XrtPDSListCreate()*で作成されたリストを破棄するには、*XrtPDSListDestroy()*を呼び出してください。

```
XrtPDSListDestroy(list);
```

リストで使用されていた全てのメモリは解放されます。

リストのコピー

既存のリストのコピーを作成するには、*XrtPDSListDuplicate()*を使用します。

```
newlist = XrtPDSListDuplicate(oldlist);
```

これはリストとそのエレメントのコピーを作成し、コピーのポインタを返します。元のリストは変更されません。

8.4.3 リストのアイテムの追加と削除

アイテムの追加

アイテムは、既存のリストの次の四つの場所に追加することができます。

- リストの最後尾
- 既存のアイテムの直前
- 既存のアイテムの直後
- ソートされた順序での該当個所

リストの最後尾にアイテムを追加するには、*XrtPDSListAppend()*を呼び出してください。

```
index = XrtPDSListAppend(list, newitem);
```

この方法は、追加の対象となるリストと追加されるアイテムの二つのパラメータが必要です。アイテムのサイズは、リストが *XrtPDSListCreate()*によって作成される時に指定されたサイズと同じにしてください。この方法は、新しいアイテムのインデックスを返します。(インデックス 0 は最初のエレメント、1 は二番目のエレメントと、続きます。)

既存のアイテムの前に新しいアイテムを追加するには、*XrtPDSListInsetBefore()*を呼び出します。

```
newindex = XrtPDSListInsertBefore(list, oldindex, newitem);
```

*XrtPDSListInsertBefore()*は、追加の対象となるリスト、既存のアイテムのインデックスと追加されるアイテムの三つのパラメータが必要です。この方法は、新しいアイテムのインデックスを返します。同様にして、*XrtPDSListInsertAfter()*は、既存のアイテムの直後に新しいアイテムを追加します。

```
newindex = XrtPDSListInsertAfter(list, oldindex, newitem);
```

この方法のパラメータと戻り値は、*XrtPDSListInsertBefore()*と同じです。

リストがソートされている場合は、*XrtPDSListAdd()*によって該当位置にアイテムを追加します。

```
index = XrtPDSListAdd(list, newitem);
```

この方法は、追加の対象となるリストと追加されるアイテムの二つのパラメータを取ります。新しいアイテムのインデックスが戻ります。(詳細は 8.4.6 を参照してください。)

アイテムの除去と削除

XrtPDSList オブジェクトでは、アイテムをリストから除去(remove)したり、削除(delete)したりすることができます。

リストからアイテムを除去するには *XrtPDSListRemove()*を呼び出してください。

```
retpos = XrtPDSListRemove(list, position);
```

この方法には、リストと除去されるアイテムの位置が渡されます。リストから全てのアイテムを除去するには、*XrtPDSListRemoveAll()*を呼びます。

```
XrtPDSListRemoveAll(list);
```

上の二つの方法では、アイテムは除去されているだけでメモリは残っています。アイテムを除去すると同時に破棄するには、*XrtPDSListDelete()*を呼び出してください。

```
retpos = XrtPDSListDestroy(list, position);
```

この方法には、*XrtPDSListRemove()*と同様に、リストと除去されるアイテムの位置が渡されます。リストの全てのアイテムを破棄するには、*XrtPDSListDestroyAll()*を呼び出してください。

```
XrtPDSListDestroyAll();
```

特定なアイテムに関連したデータ構造体が定義されており、アイテムが削除される時にその構造体も破棄するには、*XrtPDSListSetDestroyProc()*を呼び出し、破棄の手続きを定義してください。

```
XrtPDSListSetDestroyProc(list, my_destroy_proc);
```

この手続きは、*XrtPDSListDestroy()*または *XrtPDSListDestroyAll()*がコールされる度に呼び出されます。この手続きは、次のように定義してください。

```
void my_destroy_proc (
 XrtPDSList   object,
 XtPointer    item
 );
```

object はリスト、*item* は破棄されるアイテムです。

XrtPDSList オブジェクトでは、リスト上のオブジェクトを *XrtPDSListDestroyObject()*によって破棄することもできます。

```
XrtPDSList list, *objptr;

XrtPDSListDestroyObject(list, objptr);
```

ここの二つのパラメータはリストと破棄されるオブジェクトのポインタです。

リスト上のオブジェクトではないアイテムを破棄するには、*XrtPDSListDestroyXtPointer()*を呼び出してください。

```
XrtPDSList list;
XtPointer *item;

XrtPDSListDestroyXtPointer(list, item);
```

この方法には、リストと破棄されるアイテムのポインタが渡されます。

8.4.4 アイテムの検索と取得

XrtPDSList オブジェクトは、下記の検索と取得を操作する方法を提供しています。

- XrtPDSList がリストであるかを調べる
- リスト上でアイテムを検索する
- インデックス付きのエレメントまたはエレメントの配列全体を取得する
- リスト上を前方向または後ろ方向に検索する

その他の方法では、アイテムの数やそのサイズを取得することができます。

リストのチェック

*XrtPDSListIsList()*は、オブジェクトが XrtPDSList であるかどうかを調べます。

```
Boolean       is_list;
XrtPDSListobject;

is_list = XrtPDSListIsList(object);
```

*XrtPDSListISList()*は、オブジェクトが XrtPDSList オブジェクトであれば `True` を、そうでなければ `False` を返します。

アイテムを検索する

リスト上のアイテムを探し出すには、次の方法を使用してください。

- XrtPDSList の比較手続きを使用して、指定されたアイテムに適合するアイテムを探し出す *XrtPDSListFind()*
- 対応するメモリを比較しながら、指定されたアイテムと適合するアイテムを探し出す *XrtPDSListLookupItem()*

どちらの方法にも、リストと比較されるアイテムが渡されます。また、この二つの方法は、適合したアイテムがみつかった場合はそのインデックス、みつからなかった場合は XRTPDS_LIST_ITEM_NOT_FOUND を返します。

*XrtPDSListFind()*によって使用される比較手続きを指定するには、*XrtPDSListSetCompareProc()* を呼び出してください。

```
XrtPDSListSetCompareProc(list, my_compare_proc);
```

この比較手続きは、次のプロトタイプを持つように定義してください。

```
int my_compare_proc (
  XtPointer    item1,
  XtPointer    item2
  );
```

item1 と *item2* は、比較されるアイテムです。この手続きは、下記を返します。

- *item1* が *item2* より大きい場合は正の整数
- 二つのアイテムが同じであれば 0
- *item1* が *item2* より小さければ負の整数

現在使用されている比較手続きを取得するには、*XrtPDSListGetCompareProc()*を使用してください。

```
XrtPDSListItemCompareProc myproc;

myproc = XrtPDSListGetCompareProc(list);
```

返された手続きは、XrtPDSListItemCompareProc 型で、次のように定義されます。

```
typedef int (*XrtPDSListItemCompareProc) (
XtPointer item1, XtPointer item2
);
```

この比較手続きは、並べ替えにも使用できます。詳細は 8.4.6 を参照してください。

インデックスによってアイテムを取得する

位置によってアイテムを取り出すには、*XrtPDSListGetItem()*を呼び出します。

```
XtPointer     item;
XrtPDSListlist;
Int           index;

item = XrtPDSListGetItem(list, index);
```

*XrtPDSListGetItem()*には、リストと返されるアイテムの位置(0 は一番目のアイテム、1 は二番目のアイテム…)の二つのパラメータが渡されます。この位置のアイテムが返されます。位置が無効であれば、NULL が返ります。

特定な位置でのアイテムをセットするには、*XrtPDSListSetItem()*を呼び出します。

```
XrtPDSListSetItem(list, index, item);
```

この方法には、リスト、新しいアイテムのインデックスと新しいアイテム自身の三つのパラメータが渡されます。この方法は、セットが成功すれば True、指定されたインデックスが無効であれば False を返します。

リスト配列の取得

リスト･エレメントの配列全体を取得するには、*XrtPDSListGetItems()*を呼び出します。

```
XrtPDSListlist;
XtPointer     array;

array = XrtPDSListGetItems(list);
```

リストが空であれば、NULL が返ります。

アイテムの検索

指定されたアイテムを前方向に検索するには、*XrtPDSListSearchForward()*を呼び出します。

```
Int           pos;
XrtPDSListlist;
Int           start_position;
XtPointer     item;

pos = XrtPDSListSearchForward(list, start_position, item);
```

この方法は、リスト、検索の開始される位置、検索されるアイテムの三つのパラメータを必要とします。アイテムが適合すれば、そのアイテムの位置が返りますが、適合しなければ XRTPDS_LIST_ITEM_NOT_FOUND が返ります。リストを後方向に検索するには、*XrtPDSListSearchBackward()*を呼び出します。

```
pos = XrtPDSListSearchBackward(list, start_position, item);
```

パラメータと戻り値は、*XrtPDSListSearchForward()*の場合と同じです。

リスト情報を取得する

リスト上のアイテム数を取得するには、*XrtPDSListGetItemCount()*を呼び出します。

```
Int           numitems;
XrtPDSListlist;

numitems = XrtPDSListGetItemCount(list);
```

この方法には、リストが引数として渡されます。

アイテムのサイズを取得するには、*XrtPDSListGetItemSize()*を呼び出してください。

```
Int           itemsize;
XrtPDSListlist;

itemsize = XrtPDSListGetItemSize(list);
```

8.4.5 リスト比較ルーチン

XrtPDSList オブジェクトは、異なる型の値の比較に使用できるルーチンを幾つか提供しています。倍精度の浮動小数点値、単精度の浮動小数点値、整数、ストリングを比較するルーチンが定義されています。これらは全て、*XrtPDSListFind()*で使用することができます。その場合は、*XrtPDSListSetCompareProc()*に使用するルーチンを渡してください。

二つの倍精度浮動小数点値を比較するには、*XrtPDSCmpDoubles()*を呼び出してください。

```
XtPointer      double1, double2;
int            retval;

retval = XrtPDSCmpDoubles(double1, double2);
```

このルーチンは、一番目の値が二番目の値より大きい時は 1、同じであれば 0、小さければ-1 を返します。

*XrtPDSCmpFloats()*は、二つの単精度浮動小数点値を比較します。

```
XtPointer      float1, float2;
int            retval;

retval = XrtPDSCmpFloats(float1, float2);
```

戻り値は、*XrtPDSCmpDoubles()*の場合と同じです。

二つの整数を比較する場合は、*XrtPDSCmpInts()*を使用してください。

```
XtPointer      int1, int2;
int            retval;

retval = XrtPDSCmpInts(int1, int2);
```

戻り値は、*XrtPDSCmpDoubles()*の場合と同じです。

二つのストリングを比較するには、*XrtPDSCmpStrings()*を呼び出してください。

```
XtPointer      string1, string2;
int            retval;

retval = XrtPDSCmpStrings(string1, string2);
```

このルーチンは、一番目のストリングが二番目のストリングよりアルファベット順で後に来る時は 1、同じであれば 0、先に来る場合は-1 を返します。

*XrtPDSCmpString()*は大文字と小文字を区別して扱います。大文字･小文字を区別しないで比較を行うには、*XrtPDSCmpStringIgnoreCase()*を使用してください。

```
XtPointer      string1, string2;
Int            retval;

retval = XrtPDSCmpStringsIgnoreCase(string1, string2);
```

戻り値は、*XrtPDSCmpStrings()*の場合と同じです。

8.4.6 リストのソート

リスト上のアイテムをソートするには、*XrtPDSListSort()*を呼び出してください。

```
XrtPDSListSort(list);
```

*XrtPDSListSetCompareProc()*によって定義される比較手続きが、並べ替えの順序の決定に使用されます。比較手続きが指定されていない場合は、次のデフォルトの動作が適用されます。

- アイテムが 8 以下のサイズであれば、byte 順にソートされる
- アイテムが 8 より大きいサイズの場合は、比較手続きが定義されていないとソートは行われない。

尚、*XrtPDSListSetCompareProc()*に関しては、8.4.4 を参照してください。

リストがソートされているかどうかを調べるには、*XrtPDSListIsSorted()*を呼び出してください。

```
int is_sorted;

is_sorted = XrtPDSListIsSorted(list);
```

この方法は、リストがソートされている時は 0 でない値、そうでなければ 0 を返します。

エレメントの移動

エレメントの位置をリスト内で変更する時は、*XrtPDSListMoveItemsBefore()*を呼び出してください。

```
Boolean       ok;
XrtPDSListlist;
int           destination;
int           source;
int           num_items;

source = 3; /* fourth element is the first to be moved */
num_items = 3; /* move three elements */
destination = 0; /* place before first element */
ok = XrtPDSListMoveItemsBefore(list, destination, source,
  num_items);
```

*XrtPDSListMoveItemsBefore()*は、次の四つのパラメータを必要とします。

- 移動するアイテムを持つリスト
- 移動先のエレメントのインデックス(0 が一番目のエレメント、1 が二番目…)
  (移動されるエレメントのブロックは、このエレメントの直前に配置される)
- 移動されるブロックの一番目のエレメントのインデックス
- 移動されるエレメントの数

*XrtPDSListMoveItemsBefore()*は、移動が成功すると `True`、エラーが起きると `False` を返します。

8.4.7 ユーザが定義したデータ

XrtPDSList オブジェクトでは、リストに対してユーザがデータを定義することができます。データを定義するには、データのポインタを *XrtPDSListSetUserData()* に渡してください。

```
XrtPDSList     list;
XtPointer          user_data;

XrtPDSListSetUserData(list, user_data);
```

データによって使用されるメモリの解放は、ユーザ側で行ってください。

リストに対するデータを取得するには、*XrtPDSListGetUserData()* を呼び出してください。

```
user_data = XrtPDSListGetUserData(list);
```

リストにデータが定義されていなければ、この方法は NULL を返します。

8.5 手続きと方法

8.5.1 XrtPDSImage の方法

XrtPDSImageCopy()

イメージ・オブジェクトをコピーします。

```
XrtPDSImage
XrtPDSImageCopy(
  XrtPDSImage       image
  )
```

image はコピーされるイメージ。内部の pixmap がコピーされます。

XrtPDSImageCreate()

与えられた Pixmap、Display、幅、高さを使用して XrtPDSImage を作成します。

```
XrtPDSImage
XrtPDSImageCreate(
  Pixmap        pixmap,
  Display       *display,
  int           width,
  int           height
  )
```

pixmap は XrtPDSImage の Pixmap、*display* はイメージが作成される Display、*width* と *height* は作成される XrtPDSImage の幅と高さです。

XrtPDSImageDestroy()

イメージ・オブジェクトとそれに対応するメモリを破棄します。

```
void
XrtPDSImageDestroy(
  XrtPDSImage  image
  )
```

image は破棄されるイメージ。イメージの内部の pixmap が全て解放されます。

XrtPDSImageGetFilename()

イメージのファイル名を返します。このメモリは解放しないでください。

```
String
XrtPDSImageGetFilename(
  XrtPDSImage       image
  )
```

image は、名前が返されるファイルのイメージ。

XrtPDSImageGetWidth()

イメージの幅を返します。

```
int
XrtPDSImageGetWidth(
  XrtPDSImage       image
  )
```

image は、返される幅のイメージ。

XrtPDSImageGetHeight()

イメージの高さを返します。

```
int
XrtPDSImageGetHeight(
  XrtPDSImage       image
  )
```

image は、返される高さのイメージ。

XrtPDSImageGetColorDepth()

イメージの各カラー・コンポーネントに使用される bit 数を返します。(通常は 8)

```
int
XrtPDSImageGetColorDepth(
  XrtPDSImage       image
  )
```

image は、色のビット数が返されるイメージ。

XrtPDSImageGetClipMask()

イメージの clip mask を返します。clip mask が指定されていなければ XmUNSPECIFIED_PIXMAP を返します。

```
Pixmap
XrtPDSImageGetClipMask(
  XrtPDSImage       image
  )
```

image は clip mask が返されるイメージ。

XrtPDSImageGetImage()

イメージの pixmap を返します。指定されていなければ、XmUNSPECIFIED_PIXMAP を返します。

```
Pixmap
XrtPDSImageGetImage(
  XrtPDSImage       image
  )
```

image は、pixmap が返されるイメージ。

XrtPDSImageIsSet()

イメージ・オブジェクトの中の pixmap が有効であれば True を返します。

```
Boolean
XrtPDSImageIsSet(
  XrtPDSImage       image
  )
```

image は、pixmap が有効であれば True を返すイメージ。

XrtPDSImageIsClipMaskSet()

イメージ・オブジェクトの中の clip mask の pixmap が有効であれば True を返します。

```
Boolean
XrtPDSImageIsClipMaskSet(
  XrtPDSImage       image
  )
```

image は、clip mask の pixmap が有効であれば True を返すイメージ。

XrtPDSImageIsImage()

イメージが XrtPDSImage であれば True を返します。

```
Boolean
XrtPDSImageIsImage(
  XrtPDSImage       image
  )
```

image は、XrtPDSImage であれば True を返すイメージです。

8.5.2 XrtPDSList オブジェクトの方法

XrtPDSCmpDoubles()

二つの倍精度の浮動小数点数を比較します。一番目の数が二番目より大きければ 1、同じであれば 0、小さければ-1 を返します。この手続きは、*XrtPDSListSetCompareProc()*へ渡す場合に適しています。

```
int
XrtPDSCmpDoubles(
  XtPointer    double1,
  XtPointer    double2
  )
```

doubl1 と *double2* は比較される二つの数です。

XrtPDSCmpFloats()

二つの単精度の浮動小数点数を比較します。一番目の数が二番目より大きければ 1、同じであれば 0、小さければ-1 を返します。この手続きは、*XrtPDSListSetCompareProc()*へ渡す場合に適しています。

```
int
XrtPDSCmpFloats(
  XtPointer    float1,
  XtPointer    float2
  )
```

float1 と *float2* は比較される二つの数です。

XrtPDSCmpInts()

二つの整数を比較します。一番目の数が二番目より大きければ 1、同じであれば 0、小さければ-1 を返します。この手続きは、*XrtPDSListSetCompareProc()*へ渡す場合に適しています。

```
int
XrtPDSCmpInts(
  XtPointer    int1,
  XtPointer    int2
  )
```

int1 と *int2* は比較される二つの数です。

XrtPDSCmpStrings()

二つのストリングを比較します。一番目のストリングが二番目のストリングよりアルファベット順で後にあれば 1、同じであれば 0、前にあれば-1 を返します。この手続きは、*XrtPDSListSetCompareProc()*へ渡す場合に適しています。

```
int
XrtPDSCmpStrings(
  XtPointer    string1,
  XtPointer    string2
  )
```

string1 と *string2* は比較される二つのストリングです。

XrtPDSCmpStringsIgnoreCase()

二つのストリングを大文字･小文字関係なく比較します。一番目のストリングが二番目のストリングよりアルファベット順で後にあれば 1、同じであれば 0、前にあれば-1 を返します。この手続きは、*XrtPDSListSetCompareProc()*へ渡す場合に適しています。

```
int
XrtPDSCmpStringsIgnoreCase(
  XtPointer    string1,
  XtPointer    string2
  )
```

string1 と *string2* は比較される二つのストリングです。

XRTPDSListAdd()

(ソートされた)リストで該当する位置にアイテムを追加します。ソートの順序は、*XrtPDSListGetCompareProc()*によって返される手続きによって定義されます。この方法は、新しいノードの位置のインデックス、失敗時には XRTPDS_LIST_INVALID_POSITION を返します。

```
int
XrtPDSListAdd(
  XrtPDSList   list,
  XtPointer    new
  )
```

list はリスト・オブジェクト。*new* は新しいアイテムのポインタで、このサイズは*XrtPDSListGetItemSize()*によって返される値のサイズと同じです。

XrtPDSListAppend()

リストの最後にアイテムを追加します。この方法は、新しいノードの位置のインデックス、失敗時にはXRTPDS_LIST_INVALID_POSITION を返します。

```
int
XrtPDSListAppend(
  XrtPDSList   list,
  XtPointer    new
  )
```

list はリスト・オブジェクト。*new* は新しいアイテムのポインタで、このサイズは、*XrtPDSListGetItemSize()*によって返される値のサイズと同じです。

XrtPDSListCreate()

0 の長さのリストを作成し、それを返します。

```
XrtPDSList
XrtPDSListCreate(
  size_t   item_size
  )
```

item_size は、リスト上のアイテムの想定されるサイズです。

XrtPDSListDelete()

指定された位置でのアイテムを削除します。削除されたアイテムの位置を返します。

```
int
XrtPDSListDelete(
  XrtPDSList   list,
  int          position
  )
```

list はリスト・オブジェクト。*position* は削除されるアイテムの位置です。

破棄の手続きが(*XrtPDSSetDestroyProc()*によって)登録されている場合は、アイテムが削除される時にその手続きが呼び出されます。(破棄の手続きを呼ばずにアイテムを削除するには、*XrtPDSListRemove()*を使用してください。)

XrtPDSListDeleteALL()

リストからすべてのアイテムを削除します。

```
void
XrtPDSListDeleteAll(
  XrtPDSList   list,
  )
```

list は削除されるアイテムを持つリスト・オブジェクトです。

破棄の手続きが(*XrtPDSSetDestroyProc()*によって)登録されている場合は、アイテムが削除される度にその手続きが呼び出されます。(破棄の手続きを呼ばずにアイテムを削除するには、*XrtPDSListRemoveAll()*を使用してください。)

XrtPDSListDestroy()

指定されたリストを破棄し、割り当てられていたメモリを解放します。

```
void
XrtPDSListDestroy(
  XrtPDSList   list
  )
```

list は破棄されるリスト・オブジェクトです。

XrtPDSListDestroyObject()

指定されたリストのオブジェクトを破棄します。

```
void
XrtPDSListDestroyObject(
  XrtPDSList   list,
  XrtPDSList   *object
  )
```

list は破棄されるオブジェクトを持つリスト・オブジェクト、*object* は破棄されるオブジェクトです。

XrtPDSListDestroyXtPoiter()

指定されたリストのオブジェクトではないアイテムを破棄します。

```
void
XrtPDSListDestroyXtPointer(
  XrtPDSList   list,
  XtPointer    *item
  )
```

list は破棄されるアイテムを持つリスト・オブジェクト、*item* は破棄されるアイテムです。

XrtPDSListDuplicate()

リストのコピーを作成し、そのコピーを返します。新しいリスト・アイテムの破棄の手続きは NULL にセットされます。(*XrtPDSListSetDestroyProc()*でセットすることもできます。)

```
XrtPDSList
XrtPDSListDuplicate(
  XrtPDSList   list
  )
```

list はコピーされるリスト・オブジェクトです。

XrtPDSListFind()

(*XrtPDSListGetCompareProc()*によって返された手続きを使用して)指定されたアイテムに適合するリスト・アイテムを探し出します。この方法は、見つかったアイテムのインデックスを返すか、何も見つからなければXRTPDS_LIST_ITEM_NOT_FOUND を返します。

```
int
XrtPDSListFind(
  XrtPDSList   list,
  XtPointer    item
  )
```

list は検索されるリスト・オブジェクト、*item* は適合するアイテムのポインタです。リストがソートされている場合(*XrtPDSIsSorted()*が 0 でない値を返す場合)、バイナリによる検索が行われます。

リストに重複するアイテムがある場合は、最初に来るアイテムが返されます。

XrtPDSListGetCompareProc()

オブジェクトが検索とソートに使用する手続きを取得します。

```
XrtPDSListItemCompareProc
XrtPDSListGetCompareProc(
  XrtPDSList   list
  )
```

list は、比較手続きが返されるリスト・オブジェクトです。

XrtPDSListGetItem()

指定されたインデックスのアイテムのポインタを返します。インデックスが無効な場合はNULL を返します。返されたアイテムは dereference してください。

```
XtPointer
XrtPDSListGetItem(
  XrtPDSList   list,
  int          index
  )
```

list はリスト・オブジェクト、*index* は指定するアイテムのインデックス(0 は一番目のエレメント、1 は 二番目…と続きます)。

XrtPDSListGetItemCount()

リスト上のアイテムの数を取得します。

```
int
XrtPDSListGetItemCount(
  XrtPDSList   list
  )
```

list は、アイテム数がカウントされるリスト・オブジェクトです。

XrtPDSListGetItems()

アイテム配列のポインタを返します。リストが空であればNULL を返します。

```
XtPointer
XrtPDSListGetItems(
XrtPDSList list
)
```

list はリスト・オブジェクトです。

XrtPDSListGetItemSize()

各アイテムのサイズを返します。

```
int
XrtPDSListGetItemSize(
  XrtPDSList   list
  )
```

list は、サイズが返されるアイテムを持つリストです。

XrtPDSListGetUserData()

XrtPDSList オブジェクトに対するユーザ・データを取得し、それを返します。

```
XtPointer
XrtPDSListGetUserData(
  XrtPDSList   list
  )
```

list は、取得されるデータを持つリストです。

XrtPDSListInsertAfter()

指定された位置の後に新しいアイテムを挿入します。新しいノードのインデックスまたは、失敗時には XRTPDS_LIST_INVALID_POSITION を返します。

```
int
XrtPDSListInsertAfter(
  XrtPDSList   list,
  int          index,
  XtPointer    new
  )
```

list はリスト・オブジェクト、*index* は新しいアイテムが追加される位置の前のインデックス、*new* は新しいアイテムのポインタです。*index* がリスト内のエレメントの数より大きい場合は、*new* はリストの最後尾に追加されます。(*new* は、*XrtPDSListGetItemSize()*で返される値と同じサイズにしてください。)

XrtPDSListInsertBefore()

指定された位置の前に新しいアイテムを挿入します。新しいノードのインデックスまたは、失敗時には XRTPDS_LIST_INVALID_POSITION を返します。

```
int
XrtPDSListInsertBefore(
  XrtPDSList   list,
  int          index,
  XtPointer    new
  )
```

list はリスト・オブジェクト、*index* は新しいアイテムが追加される位置の後のインデックス、*new* は新しいアイテムのポインタです。*index* がリスト内のエレメントの数より大きい場合は、*new* はリストの最後に追加されます。(*new* は *XrtPDSListGetItemSize()*によって返される値と同じサイズにしてください。)

XrtPDSListIsList()

渡される XrtPDSList が XrtPDSList オブジェクトであれば True を返します。

```
Boolean
XrtPDSListIsList(
  XrtPDSList   object
  )
```

object はチェックされるオブジェクトです。

XrtPDSListIsSorted()

リストが現在ソートされている状態であるかを示します。ソートされていれば 0 ではない値を返します。

```
int
XrtPDSListIsSorted(
  XrtPDSList   list
  )
```

list はチェックされるリスト・オブジェクトです。

XrtPDSListLookupItem()

指定されたリストでアイテムを探し出します。このルーチンは、*XrtPDSListGetCompareProc()* によって返された手続きは使用せず、メモリを厳密に調べていきます。この方法は、適合されたアイテムの位置または、XRTPDS_LIST_ITEM_NOT_FOUND を返します。

```
int
XrtPDSListLookupItem(
XrtPDSListlist,
XtPointer      item
)
```

list は検索されるリスト、*item* は探し出されるアイテムのポインタです。

XrtPDSListMoveItemsAfter()

指定された数のアイテムを移動します。これらのアイテムは、指定された位置の後に配置されます。

```
Boolean
XrtPDSListMoveItemsAfter(
  XrtPDSList   list,
  int          destination,
  int          source,
  int          num_items_to_move
)
```

list はリスト・オブジェクト、*destination* は移動先で、移動されるアイテムはこの後に配置されます。*source* は移動される最初のアイテムの位置、*num_items_to_move* は移動されるアイテムの数です。

XrtPDSListMoveItemsBefore()

指定された数のアイテムを移動します。これらのアイテムは、指定された位置の前に配置されます。

```
Boolean
XrtPDSListMoveItemsBefore(
  XrtPDSList   list,
  int          destination,
  int          source,
  int          num_items_to_move
  )
```

list はリスト・オブジェクト、*destination* は移動先で、移動されるアイテムはこの前に配置されます。*source* は移動される最初のアイテムの位置、*num_items_to_move* は移動されるアイテムの数です。

XrtPDSListRemove()

指定された位置のアイテムを除去します。除去されたアイテムの位置を返します。

```
int
XrtPDSListRemove(
  XrtPDSList   list,
  int          position
  )
```

list はリスト・オブジェクト、*position* は除去されるアイテムの位置です。

(*XrtPDSSetDestroyProc()*によって登録される)破棄の手続きは、アイテムが除去される時には呼び出されません。アイテムを削除し、破棄の手続きを呼び出すには、*XrtPDSListDelete()*を使用してください。

XrtPDSListRemoveAll()

リストから全てのアイテムを除去します。

```
void
XrtPDSListRemoveAll(
  XrtPDSList   list
  )
```

list は、除去されるアイテムを持つリスト・オブジェクトです。

(*XrtPDSSetDestroyProc()*によって登録される)破棄の手続きは、アイテムが除去される時には呼び出されません。各アイテムを削除し、破棄の手続きを呼び出すには、*XrtPDSListDeleteAll()*を使用してください。

XrtPDSListSearchForward()

指定された位置から前方向にリスト上を検索し、条件と適合するアイテムを探し出します。*XrtPDSListGetCompareProc()*によって返される手続きが使用されます。適合したアイテムの位置または、XRTPDS_LIST_ITEM_NOT_FOUND を返します。

```
int
XrtPDSListSearchForward(
  XrtPDSList    list,
  int           start_position,
  XtPointer     item
  )
```

list は検索されるリスト、*start_position* は比較される最初のアイテムの位置、*item* は適合されるアイテムのポインタです。

XrtPDSListSearchReverse()

指定された位置から後方向にリスト上を検索し、条件と適合するアイテムを探し出します。*XrtPDSListGetCompareProc()*によって返される手続きが使用されます。適合したアイテムの位置または、XRTPDS_LIST_ITEM_NOT_FOUND を返します。

```
int
XrtPDSListSearchReverse(
  XrtPDSList    list,
  int           start_position,
  XtPointer     item
  )
```

list は検索されるリスト、*start_position* は比較される最初のアイテムの位置、*item* は適合されるアイテムのポインタです。

XrtPDSListSetCompareProc()

リスト･オブジェクトがソートに使用する手続きを指定します。

```
void
XrtPDSListSetCompareProc(
  XrtPDSList                    list,
  XrtPDSListItemCompareProc compare_proc
  )
```

list は比較手続きがセットされるリスト･オブジェクト、*compare_proc* は呼び出される手続きです。

比較手続きは、次のようにして定義されます。

```
int (*XrtPDSListItemCompareProc) (
  XtPointer     item1,
  XtPointer     item2
  );
```

item1 と *item2* は比較される二つのアイテムです。比較手続きは、下記のうちどれかを返します。

- *item1* が *item2* より大きい場合は正の整数
- 二つのアイテムが同じであれば0
- *item1* が *item2* より小さければ負の整数

XrtPDSListSetDestroyProc()

リスト・オブジェクトがアイテムを破棄する時に使用する手続きを指定します。

```
void
XrtPDSListSetDestroyProc(
  XrtPDSList                    list,
  XrtPDSListItemDestroyProc destroy_proc
  )
```

list はアイテム破棄の手続きがセットされるリスト・オブジェクト。*destroy_proc* は呼び出される手続きです。

破棄の手続きは、次のようにして定義されます。

```
void (*XrtPDSListItemDestroyProc) (
  XrtPDSList object,
  XtPointer        item,
  XtPointer        user_data
  );
```

object は破棄されるアイテムを持つリスト・オブジェクト、*item* は破棄されるアイテムです。*user_data* は(*XrtPDSListSetUserData()*でセットされる)データです。

XrtPDSListSetItem()

アイテムをセットします。指定されたインデックスが無効な場合は `False` を返します。

```
Boolean
XrtPDSListSetItem(
  XrtPDSList   list,
  int          index,
  XtPointer    item
  )
```

list はリスト・オブジェクト、*index* はセットされるアイテムの位置、*item* はセットされるアイテムのポインタです。

XrtPDSListSetUserData()

XrtPDSList オブジェクトに対するユーザ・データをセットします。

```
void
XrtPDSListSetUserData(
  XrtPDSList   list,
  XtPointer    user_data
  )
```

list はリスト・オブジェクト、*user_data* はユーザ・データです。

XrtPDSListSort()

アイテムをソートします。リストが既にソートされている場合は、何もしません。

```
void
XrtPDSListSort(
  XrtPDSList   list
  )
```

list はソートされるリストです。

Part II

Reference Appendices

A

リソースの一覧

ウィジェットの概要
XRT/graph のリソース
継承されるリソース

XRT/graph の全てのリソースのアルファベット順の一覧です。リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C** は、そのリソースが create(例: *XtCreateWidget()*)で定義でき、**S** は set(例: *XtSetValues()*)でき、**G** は get(例: *XtGetValues()*)で使用できることを意味します。

A.1 ウィジェットの概要

インクルード・ファイル:	*<Xm/XrtGraph.h>*
クラス名	XtXrtGraph
クラス階層	Core → XmPrimitive → XtXrtGraph
クラス・ポインタ	xtXrtGraphWidgetClass
インスタンスの生成	`widget = XtCreateWidget(name, xtXrtGraphWidgetClass ...)`
関数/マクロ	*XtCreateXrtGraph()*、*XtIsXrtGraph()*

A.2 XRT/graph のリソース

XtNxrtAngleUnit　XrtAngleUnit　XRT_ANGLE_DEGREES　CSG

データの X 値と Y 軸のリソースを解釈する時のユニットを指定します。主としてポーラー・グラフに使用されますが、レーダー、filled レーダー・グラフにも使用されます。有効な値は、`XRT_ANGLE_DEGREES`、`XRT_ANGLE_RADIANS`、`XRT_ANGLE_GRADS` です。

XtNxrtAxisBoundingBox Boolean False CSG

True の場合、グラフ・データの領域(軸、表示されているデータ、グリッド線)を囲むボックスが描かれます。ポーラー、円グラフには使用されません。

XtNxrtAxisFont XmFontList “fixed” CSG

軸の数値ラベル、注釈、タイトルに使用されるフォントを指定します。テキスト・オブジェクトでフォントが指定されていない場合は、このフォントが使用されます。

XtNxrtBackgroundColor char* "White" CSG

ウィンドウの背景の色を指定します。ストリングは、色名か#で始まる RGB(赤、緑、青)構成の二桁の 16 進数です。色の指定については、セクション 2.8 をご覧ください。この色は、グラフ、データ領域、ヘッダー、フッター、凡例のデフォルトの背景色として継承されます。

モノクロのサーバでは、黒以外の色を指定しますと白に表示されます。

XtNxrtBarClusterOverlap int 100 CSG

クラスタ内の棒の重なり合いを棒の幅の割合(%)で指定します。0 に設定すると全ての棒が完全に重なります。100 では棒が隣り合って並びます。この値の範囲は 0 から 200 です。棒の実際の幅は、ウィンドウのサイズやその他の要因に依存し、XRT/graph が実行時に決定します。積上げ棒では、一つのクラスタにつき一本の棒なので、XtNxrtType が XRT_TYPE_STACKING_BAR の場合このリソースは関係ありません。

XtNxrtBarClusterWidth int 50 CSG

クラスタ内で棒が占めるスペースを、クラスタに割り当てられているスペースに対する割合で指定します。値の範囲は 0 から 100 です。100 に設定すると棒が隣のクラスタの棒と接します。

XtNxrtBorder XrtBorder(enum) XRT_BORDER_3D_OUT CSG

グラフを囲む枠線のスタイルを指定します。有効な値は、XRT_BORDER_NONE、XRT_BORDER_PLAIN、XRT_BORDER_3D_IN、XRT_BORDER_3D_OUT、XRT_BORDER_ETCHED_IN、XRT_BORDER_ETCHED_OUT 、 XRT_BORDER_FRAME_IN 、 XRT_BORDER_FRAME_OUT 、XRT_BORDER_BEVEL、XRT_BORDER_SHADOW。枠の幅は XmNshadowThickness で指定します。

XtNxrtBubbleMethod XrtBubbleMethod (enum) XRT_BUBBLE_METHOD_DIAMETER CSG

バブル・チャートの作成を指定します。値は、XRT_BUBBLE_METHOD_DIAMETER か XRT_BUBBLE_METHOD_AREA のどちらかです。XRT_BUBBLE_METHOD_DIAMETER はポイント・シンボルの直径によって、XRT_BUBBLE_METHOD_AREA はポイント・シンボルが占める領域によってデータを表します。

XtNxrtBubbleMax float 20.0 CSG
XtNxrtBubbleMin float 5.0 CSG

ポイント・シンボルの直径/領域の最小と最大をグラフの直径/領域に対する割合として指定します。XtNxrtBubbleMin の値が XtNxrtBubbleMax の値より大きい場合は、ポイント・シンボルの直径/領域は、バブル値が増加するにつれて減少していきます。ポイント・シンボルの直径/領域は、バブル値の範囲に依存しますので、バブル値は必ずしも全てが正の値または負の値である必要はありません。

注意: float 型のリソースをセットする時は、*XrtFloatToArgVal()*を使用してください。

XtNxrtCandleComplex Boolean False CSG

グラフ・タイプが XRT_TYPE_CANDLE の場合にのみ参照されます。False の場合は、各 logical(論理的)セットの最初のセットのデータ・スタイルを使用してキャンドルの表示をコントロールします。True の場合は、四つ全てのセットのデータ・スタイルが使用されます。

XtNxrtData XrtDataHandle none CSG
XtNxrtData2 XrtDataHandle none CSG

表示されるデータを指定します。データ・オブジェクトは、*XrtDataCreate()* または *XrtDataCreateFromFile()* を使用して割り当てます。XtNxrtData2 は複合グラフで二番目のグラフのデータを指定します。データのリソースは、他のリソースと異なり、ウィジェットによってコピーされません。これらのデータ・オブジェクトの割り当てや、割り当て解除は、アプリケーションの役割です。

XtNxrtDataAreaBackgroundColor char* NULL CSG

グラフのデータ領域の背景色を指定します。XtNxrtDataAreaBackGroundColor と XrtNxrtDataAreaImage が両方とも NULL の場合は、無色です。ストリングは、色名か# で始まる RGB(赤、緑、青)構成の二桁の 16 進数です。色の指定については、セクション 2.8 をご覧ください。このリソースは、円グラフでは無視されます。

モノクロのサーバでは、黒以外の色を指定しますと白に表示されます。

XtNxrtDataAreaForegroundColor char* NULL CSG

棒、塗りつぶされる領域、円のスライスを囲む線(XtNxrtGraphShowOutlines が True)の色を指定します。NULL の場合は、グラフの前景色が指定されます。ストリングは、色名か#で始まる RGB(赤、緑、青)構成の二桁の 16 進数です。色の指定については、セクション 2.8 をご覧ください。

モノクロのサーバでは、白以外の色を指定しますと黒に表示されます。

XtNxrtDataAreaImage XrtImage NULL CSG

グラフのデータ領域の背景に表示するイメージを指定します。

XtNxrtDataAreaImageLayout XrtImageLayout XRT_IMAGELAYOUT_CENTERED CSG

背景のイメージの位置とサイズを指定します。有効な値は、XRT_IMAGELAYOUT_CENTERED か XRT_IMAGELAYOUT_TILED です。

XtNxrtDataStyles	XrtDataStyle**	dynamic	CSG
XtNxrtDataStyles2	XrtDataStyle**	dynamic	CSG

`XrtDataStyle` 構造体は、各データ・セットの線や塗りつぶしのパターン、色、線幅、ポイントのスタイルやサイズ、色を定義します。XRT/graph は、グラフ・オブジェクトの生存期間中、描画される時にのみデフォルトの `XrtData` スタイルを作成します。

`XtNxrtDataStyles` でどのデータ・スタイルが現在使用されているか確認したり、スタイルを定義したりすることができます。スタイルのリストは、`NULL` ポインタで終了します。

*XtGetValues()*と共に使用された場合は、`XrtDataStyle` 構造体の配列のポインタが返されます。配列の長さは、対応するデータ・オブジェクトの現在のセット数か、プログラムによって指定されたデータ・スタイルの一番高いインデックスのどちらか大きい方です。

`XtNxrtDataStyles2` は、複合グラフの二番目のデータ・スタイルにアクセスする時に使用します。

XtNxrtDataStylesUseDefault	Boolean	True	CSG
XtNxrtDataStyles2UseDefault	Boolean	True	CSG

`True` の場合、データ・スタイル値は実行時に XRT/graph によって決定されます。コンパイル時にサーバのタイプ(カラーかモノクロ)が不明で、サーバ別のデータ・スタイル設定を避ける時に便利です。データ・スタイルが明示的に設定されている場合は `False` にセットされます。

`XtNxrtDataStyles2UseDefault` は、複合グラフの二番目のグラフのデータ・スタイルに適用されます。

XtNxrtDebug	Boolean	False	CSG

警告メッセージを *stderr* に送るかどうか指定します。メッセージ・テキストのカスタマイズには Xt Error Database を使用できます。`False` にセットされると、リソース変換に関するエラーだけが出力されます。

XtNxrtDoubleBuffer	Boolean	True	CSG

`True` の場合、グラフのアップデートはオフ・スクリーンで実行され、次に表示領域にコピーされます。この機能は画面のちらつきを減少させます。`False` に設定すると、グラフのアップデートは直接表示され、その結果、X サーバで使用されるメモリは少なくなります。

XtNxrtExtraDefaultDataStyles	Boolean	False	CSG

追加の八つのデータ・スタイルを使用できるようにするかどうかを指定します。バージョンの互換性を重視する場合は、使用できないようにしてください。create の後にこのリソースを変更しても効果はありません。

XtNxrtExposeCallback	XtCallbackList	NULL	CSG

XRT/graph がグラフ(の一部または全て)を再描画した時、アプリケーションに通知するために呼ばれるコールバックのリストを指定します。`XrtCallbackStruct` のポインタが各コールバックに渡されます。*XtAddCallback()*を使用してリストにコールバックを加えてください。

コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget       graph,
  caddr_t      client_data,
  XrtCallbackStruct*call_data
  )
```

XrtCallbackStruct は次のように定義されます。

```
typedef struct {
  int     reason; /* Read-only */
  XEvent  *event; /* Read-only */
  Window  window; /* Read-only */
  Region  region; /* Read-only */
  int     change; /* Read-only */
} XrtCallbackStruct;
```

reason は常に XmCR_EXPOSE です。*region* は再描画を必要とするウィジェットの領域、(何も再描画されなければ NULL)、*change* はウィジェットのどの部分が実際に再描画されたかを記述し、次の値で構成されます。

```
#define XRT_GRAPH_CHANGED   0x01
#define XRT_HEADER_CHANGED  0x02
#define XRT_FOOTER_CHANGED  0x04
#define XRT_LEGEND_CHANGED  0x08
#define XRT_DATA_CHANGED    0x09 /* graph and legend */
#define XRT_MARKER_CHANGED  0x10
#define XRT_ALL_CHANGED0xFF
```

データの変更は、グラフと凡例の再描画が必要です。また、ウィジェットの *XtSetValues()*の呼び出しが expose コールバック内からであると、無限ループを避けるために、expose コールバックのリストは呼び出されません。

XtNxrtFooterAdjust　XrtAdjust(enum)　XRT_ADJUST_CENTER　CSG

フッター領域に複数行のテキストをどのように行揃えするか指定します。XRT_ADJUST_CENTER、XRT_ADJUST_LEFT、XRT_ADJUST_RIGHT のどれかを指定します。

XtNxrtFooterBackgroundColor　char*　NULL　CSG

フッターの背景色を指定します。XtNxrtFooterBackgroundColor と XtNxrtFooterImage が両方とも NULL であれは無色になります。ストリングは色名か#で始まる RGB(赤、緑、青)構成の二桁の 16 進数です。色の指定についてはセクション 2.8 をご覧ください。

モノクロのサーバでは、黒以外の色を指定しますと白に表示されます。

XtNxrtFooterBorder　XrtBorder(enum)　XRT_BORDER_NONE　CSG

フッター領域の枠線を指定します。有効なスタイルは、XRT_BORDER_NONE、XRT_BORDER_PLAIN、XRT_BORDER_3D_IN、XRT_BORDER_3D_OUT、XRT_BORDER_ETCHED_IN、XRT_BORDER_ETCHED_OUT、XRT_BORDER_FRAME_IN、XRT_BORDER_FRAME_OUT、XRT_BORDER_BEVEL、XRT_BORDER_SHADOW です。

XtNxrtFooterBorderWidth	Dimension	2	CSG

フッター領域の枠線の幅を pixel で指定します。範囲は 0 から 20 です。

XtNxrtFooterFont	XmFontList	“fixed”	CSG

フッター・ストリングのフォントを指定します。

XtNxrtFooterForegroundColor	char*	NULL	CSG

フッター領域の前景色を指定します。NULL であれば、ウィンドウの前景色が使用されます。ストリングは色名か#で始まる RGB(赤、緑、青)構成の二桁の 16 進数です。色の指定についてはセクション 2.8 をご覧ください。

モノクロのサーバでは、白以外の色を指定しますと黒に表示されます。

XtNxrtFooterImage	XrtImage	NULL	CSG

フッター領域の背景に表示するイメージのファイル名を指定します。

XtNxrtFooterImageLayout	XrtImageLayout	XRT_IMAGELAYOUT_CENTERED	CSG

背景のイメージの位置とレイアウトを指定します。有効な値は、XRT_IMAGELAYOUT_CENTERED か XRT_IMAGELAYOUT_TILED のどちらかです。

XtNxrtFooterImageMinimumSize	Boolean	False	CSG

イメージのサイズに対応するラベルの最小のサイズを指定します。テキストでないイメージ(グラフ領域の背景のイメージ等)には使用できません。

XtNxrtFooterStrings	char**	none	CSG

フッター領域に表示されるテキストを指定します。

XtNxrtFooterHeight	Dimension	dynamic	G
XtNxrtFooterWidth	Dimension	dynamic	G

フッター領域の幅と高さ(pixel)を持っています。

XtNxrtFooterX	Position	**グラフの下に中央揃え**	CSG
XtNxrtFooterY	Position	**グラフの下に中央揃え**	CSG

フッター領域の左上角の X、Y 座標を指定します。この設定の副作用で、対応する UseDefault リソースが False にセットされます。

XtNxrtFooterXUseDefault	Boolean	True	CSG
XtNxrtFooterYUseDefault	Boolean	True	CSG

True であれば、XtNxrtFooterX と XtNxrtFooterY 値が XRT/graph によって決定されます。XtNxrtFooterX と XtNxrtFooterY の値が設定されている場合は、XRT/graph はこれらのリソースを False にセットします。値が提供されていないとこれらのリソースは False にできません。

XtNxrtForegroundColor char* "Black" CSG

ヘッダー、フッター、凡例のテキスト、軸の表示に使用されるウィンドウの前景色を指定します。ストリングは色名か#で始まる RGB(赤、緑、青)構成の二桁の 16 進数です。色の指定についてはセクション 2.8 をご覧ください。この色は、ヘッダー、フッター、凡例、軸のデフォルトの前景色として継承されます。

モノクロのサーバでは、白以外の色を指定しますと黒に表示されます。

XtNxrtFrontDataset int XRT_DATASET1 CSG

複合グラフでどちらのデータセットが前面に表示されるか指定します。有効な値は XRT_DATASET1 か XRT_DATASET2 です。

XtNxrtGraph3dShading XrtShading XRT_SHADING_COLOR CSG

グラフが 3D 表示になっている時[1]、影の部分をどのように表示するか指定します。XRT_SHADING_COLOR の場合は、各棒または円のスタイスに対して新たな色を 2 色作成し、XRT_SHADING_DITHERED の場合は、他の色を割り当てずに、既存の色だけを使用して影の濃淡を表示します。XRT_SHADING_NONE の場合は、影は表示されません。

XtNxrtGraphBackgroundColor char* NULL CSG

グラフ領域の背景色を指定します。XtNxrtGraphBackgroundColor と XtNxrtGraphImage が両方とも NULL であれば無色です。ストリングは色名か#で始まる RGB(赤、緑、青)構成の二桁の 16 進数です。色の指定についてはセクション 2.8 をご覧ください。

モノクロのサーバでは、黒以外の色を指定しますと白に表示されます。

XtNxrtGraphBorder XrtBorder(enum) XRT_BORDER_NONE CSG

グラフ領域を囲む線のスタイルを指定します。有効なスタイルは XRT_BORDER_NONE、XRT_BORDER_PLAIN、XRT_BORDER_3D_IN、XRT_BORDER_3D_OUT、XRT_BORDER_ETCHED_IN、XRT_BORDER_ETCHED_OUT、XRT_BORDER_FRAME_IN、XRT_BORDER_FRAME_OUT、XRT_BORDER_BEVEL、XRT_BORDER_SHADOW、XRT_BORDER_PLAIN です。

XtNxrtGraphBorderWidth Dimension 2 CSG

グラフ領域の枠線の幅を pixel で指定します。範囲は 0 から 20 です。

XtNxrtGraphDepth int 0 CSG

3D 表示にした時のグラフの奥行きをグラフの幅に対する割合で指定します。3D 表示するには、グラフの奥行きと、傾きか回転のどちらかを非 0 の値にしてください。範囲は 0 から 500 です。3D 表示できるグラフは、棒、積み上げ棒、円グラフだけです。

[1] 3D 表示は、XtNxrtGraphDepth、XtNxrtGraphInclination、XtNxrtGraphRotation でコントロールされます。

XtNxrtGraphForegroundColor　char*　NULL　CSG

グラフ領域の前景色を指定します。NULL であれば、ウィンドウの前景色が使用されます。ストリングは色名か#で始まる RGB(赤、緑、青)構成の二桁の 16 進数です。色の指定についてはセクション 2.8 をご覧ください。

モノクロのサーバでは、白以外の色を指定しますと黒に表示されます。

XtNxrtGraphImage　XrtImage　NULL　CSG

グラフ領域の背景に表示するイメージのファイル名を指定します。

XtNxrtGraphImageLayout　XrtImageLayout　XRT_IMAGELAYOUT_CENTERED　CSG

背景のイメージの位置とサイズを指定します。有効な値は XRT_IMAGELAYOUT_CENTERED か XRT_IMAGELAYOUT_TILED です。

XtNxrtGraphInclination　int　0　CSG

3D 表示した時、X 軸の上の視点から見た傾きを指定します。3D 表示にするには、グラフの奥行きと傾きか回転のどちらかを非 0 の値にしてください。範囲は-45 から 45 です。3D 表示できるグラフは、棒、積み上げ棒、円グラフだけです。

XtNxrtGraphMarginBottom　Position　dynamic　CSG

グラフ領域の下部からデータ領域が表示されるまでの距離(pixel)を指定します。XtNxrtGraphMarginBottomUseDefault が True であれば、XRT/graph は、軸の注釈やタイトルに必要なスペースに基づいてこの値を計算します。このリソースを設定すると、副作用で対応する UseDefault リソースが False にセットされます。このリソースは、XRT_MIN_MARGIN と XRT_MAX_MARGIN の間でなければなりません。

XtNxrtGraphMarginBottomUseDefault　Boolean　True　CSG

True であれば、XtNxrtGraphMarginBottom は XRT/graph によって決定されます。XtNxrtGraphMarginBottom を設定すると、このリソースは False になります。XtNxrtGraphMarginBottom の値が提供されていないと、このリソースは False にできません。

XtNxrtGraphMarginLeft　Position　dynamic　CSG

グラフ領域の左からデータ領域が表示されるまでの距離(pixel)を指定します。XtNxrtGraphMarginLeftUseDefault が True であれば、XRT/graph は、軸の注釈やタイトルに必要なスペースに基づいてこの値を計算します。このリソースを設定すると、副作用で対応する UseDefault リソースが False にセットされます。

XtNxrtGraphMarginLeftUseDefault　Boolean　True　CSG

True であれば、XtNxrtGraphMarginLeft は XRT/graph によって決定されます。XtNxrtGraphMarginLeft を設定すると、このリソースは False になります。XtNxrtGraphMarginLeft の値が提供されていないと、このリソースは False にできません。

XtNxrtGraphMarginRight Position dynamic CSG

グラフ領域の右からデータ領域が表示されるまでの距離(pixel)を指定します。XtNxrtGraphMarginRightUseDefault が True であれば、XRT/graph は、軸の注釈やタイトルに必要なスペースに基づいてこの値を計算します。このリソースを設定すると、副作用で対応する UseDefault リソースが False にセットされます。

XtNxrtGraphMarginRightUseDefault Boolean True CSG

True であれば、XtNxrtGraphMarginRight は XRT/graph によって決定されます。XtNxrtGraphMarginRight を設定すると、このリソースは False になります。XtNxrtGraphMarginRight の値が提供されていないと、このリソースは False にできません。

XtNxrtGraphMarginTop Position dynamic CSG

グラフ領域の上部からデータ領域が表示されるまでの距離(pixel)を指定します。XtNxrtGraphMarginTopUseDefault が True であれば、XRT/graph は、軸の注釈やタイトルに必要なスペースに基づいてこの値を計算します。このリソースを設定すると、副作用で対応する UseDefault リソースが False にセットされます。

XtNxrtGraphMarginTopUseDefault Boolean True CSG

True であれば、XtNxrtGraphMarginTop は XRT/graph によって決定されます。XtNxrtGraphMarginTop を設定すると、このリソースは False になります。XtNxrtGraphMarginTop の値が提供されていないと、このリソースは False にできません。

XtNxrtGraphRotation int 0 CSG

3D 表示した時、Y 軸の右の視点からの回転角度を指定します。3D 表示にするには、グラフの奥行きと傾きか回転のどちらかは非 0 の値にしてください。範囲は-45 から 45 です。3D 表示できるグラフは、棒、積み上げ棒、円グラフだけです。

XtNxrtGraphShowOutlines Boolean True CSG

棒、塗りつぶされた領域、円のスライスの周囲に線を描くかどうか指定します。True の場合は、(XtNxrtDataAreaForegroundColor で指定された色を使用して)線が描かれます。

XtNxrtGraphHeight Dimension dynamic CSG
XtNxrtGraphWidth Dimension dynamic CSG

グラフ領域の幅と高さ(pixel)を指定します。これらのリソース設定すると、副作用で対応する UseDefault リソースが False にセットされます。

軸と注釈はグラフ領域に含まれますが、ヘッダー、フッターと凡例は含まれません。グラフ・ウィジェット全体のサイズを変更する場合は、XtNwidth と XtNheight を使用してください。

XtNxrtGraphWidthUseDefault Boolean True CSG
XtNxrtGraphHeightUseDefault Boolean True CSG

True であれば、XtNxrtGraphWidth と XtNxrtGraphHeight の値は XRT/graph によって決定されます。XtNxrtGraphWidth と XtNxrtGraphHeight の値が設定されている時は、XRT/graph はこれらを False にします。XtNxrtGraphWidth と XtNxrtGraphHeight の値が提供されていないと、これらのリソースは False にできません。

XtNxrtGraphX	Position	dynamic	CSG
XtNxrtGraphY	Position	dynamic	CSG

グラフ領域の左上角の(X、Y)座標を指定します。これらのリソースを設定すると、副作用で、対応する UseDefault リソースは False にセットされます。

XtNxrtGraphXUseDefault	Boolean	True	CSG
XtNxrtGraphYUseDefault	Boolean	True	CSG

True の場合、XtNxrtGraphX と XtnxrtGraphY は XRT/graph によって決定されます。XtNxrtGraphX と XtNxrtGraphY の値が設定されている時は、XRT/graph はこれらを False にします。XtNxrtGraphX と XtNxrtGraphY の値が提供されていないと、これらのリソースは False にできません。

XtNxrtHeaderAdjust	XrtAdjust(enum)	XRT_ADJUST_CENTER	CSG

複数行のヘッダーテキストがどのように行揃えされるか指定します。XRT_ADJUST_CENTER、XRT_ADJUST_LEFT、XRT_ADJUST_RIGHT のどれかです。

XtNxrtHeaderBackgroundColor	char*	NULL	CSG

ヘッダー領域の背景色を指定します。XtNxrtHeaderBackgroundColor と XtNxrtHeaderImage が両方とも NULL であれば無色です。ストリングは色名か#で始まる RGB(赤、緑、青)構成の二桁の 16 進数です。色の指定についてはセクション 2.8 をご覧ください。

モノクロのサーバでは、黒以外の色を指定しますと白に表示されます。

XtNxrtHeaderBorder	XrtBorder(enum)	XRT_BORDER_NONE	CSG

ヘッダー領域の枠線のスタイルを指定します。有効なスタイルは XRT_BORDER_NONE、XRT_BORDER_PLAIN 、 XRT_BORDER_3D_IN 、 XRT_BORDER_3D_OUT 、XRT_BORDER_ETCHED_IN 、 XRT_BORDER_ETCHED_OUT 、 XRT_BORDER_FRAME_IN 、XRT_BORDER_FRAME_OUT、XRT_BORDER_BEVEL、XRT_BORDER_SHADOW、です。

XtNxrtHeaderBorderWidth	Dimension	2	CSG

ヘッダー領域の枠線幅を pixel で指定します。範囲は 0 から 20 です。

XtNxrtHeaderFont	XmFontList	“fixed”	CSG

ヘッダー・ストリングに使用されるフォントを指定します。

XtNxrtHeaderForegroundColor	char*	NULL	CSG

ヘッダー領域の前景色を指定します。NULL であればウィンドウの前景色が使用されます。ストリングは色名か#で始まる RGB(赤、緑、青)構成の二桁の 16 進数です。色の指定についてはセクション 2.8 をご覧ください。

モノクロのサーバでは、白以外の色を指定しますと黒に表示されます。

XtNxrtHeaderImage	XrtImage	NULL	CSG

ヘッダー領域の背景に表示するイメージのファイル名を指定します。

XtNxrtHeaderImageLayout	XrtImageLayout	XRT_IMAGELAYOUT_CENTERED	CSG

背景のイメージの位置とサイズを指定します。有効な値は `XRT_IMAGELAYOUT_CENTERED` か `XRT_IMAGELAYOUT_TILED` です。

XtNxrtHeaderImageMinimumSize	Boolean	False	CSG

イメージのサイズに対応するラベルの最小サイズを指定します。テキストではなイメージ(グラフ領域の背景のイメージ等)には使用できません。

XtNxrtHeaderStrings	char**	none	CSG

ヘッダー領域に表示されるテキストを指定します。

XtNxrtHeaderHeight	Dimension	dynamic	G
XtNxrtHeaderWidth	Dimension	dynamic	G

ヘッダー領域の幅と高さ(pixel)を持っています。

XtNxrtHeaderX	Position	**グラフの上に中央揃え**	CSG
XtNxrtHeaderY	Position	**グラフの上に中央揃え**	CSG

ヘッダー領域の左上角の(X、Y)座標を指定します。これらのリソースを設定すると、副作用で、対応する `UseDefault` リソースは `False` になります。

XtNxrtHeaderXUseDefault	Boolean	True	CSG
XtNxrtHeaderYUseDefualt	Boolean	True	CSG

`True` の場合、`XtNxrtHeaderX` と `XtNxrtHeaderY` の値は XRT/graph によって決定されます。`XtNxrtHeaderX` と `XtNxrtHeaderY` の値が与えられている場合は XRT/graph はこれらを `False` にします。`XtNxrtHeaderX` と `XtNxrtHeaderY` の値が提供されていないと、`False` に指定できません。

XtNxrtHiLoCloseShow	Boolean	True	CSG

グラフ・タイプが `XRT_TYPE_HILO_OPEN_CLOSE` の時、close の tick マークを表示するかどうか指定します。

XtNxrtHiLoOpenCloseFullWidth	Boolean	False	CSG

グラフ・タイプが `XRT_TYPE_HILO_OPEN_CLOSE` の時、open/close の tick マークを幅全体に渡って描画するかどうか指定します。エラー・バー・チャートを作成する時に便利です。

XtNxrtHiLoOpenShow	Boolean	True	CSG

グラフ・タイプが `XRT_TYPE_HILO_OPEN_CLOSE` の時、open の tick マークを表示するかどうか指定します。

XtNxrtImage	XrtImage	NULL	CSG

ウィジェットの背景に表示するイメージのファイル名を指定します。

XtNxrtImageLayout	XrtImageLayout	XRT_IMAGELAYOUT_CENTERED	CSG

背景のイメージの位置とサイズを指定します。有効な値は、XRT_IMAGELAYOUT_CENTERED か XRTIMAGELAYOUT_TILED です。

XtNxrtInvertOrientation	Boolean	False	CSG

X と Y を入れ替えます。True であれば、X 値が縦軸に、Y 値が横軸に表示されます。棒グラフ、積上げ棒グラフは横に表示されます。円グラフ、ポーラー、レーダー、filled レーダーには影響しません。

XtNxrtIsStacked	Boolean	False	CSG
XtNxrtIsStacked2	Boolean	False	CSG

グラフをスタッキング(積上げ)するかどうかを指定します。スタッキングが可能なグラフは、線、面、ポーラー、レーダー、filled レーダーです。棒グラフには、既に積上げ棒グラフという種類があるので、このリソースは使用できません。(注意: スタッキングには配列型のデータが必要です。)

XtNxrtLegendAnchor	XrtAnchor(enum)	XRT_ANCHOR_EAST	CSG

凡例の位置を指定します。有効な値は XRT_ANCHOR_NORTH、XRT_ANCHOR_SOUTH、XRT_ANCHOR_EAST 、 XRT_ANCHOR_WEST 、 XRT_ANCHOR_NORTHWEST 、 XRT_ANCHOR_NORTHEAST、XRT_ANCHOR_SOUTHWEST、XRT_ANCHOR_SOUTHEAST です。

XtNxrtLegendBackgroundColor	char*	NULL	CSG

凡例領域の背景色を指定します。XtNxrtLegendBackgroundColor と XtNxrtLegendImage の両方が NULL であれば無色です。ストリングは色名か#で始まる RGB(赤、緑、青)構成の二桁の 16 進数です。色の指定についてはセクション 2.8 をご覧ください。

モノクロのサーバでは、黒以外の色を指定しますと白に表示されます。

XtNxrtLegendBorder	XrtBorder(enum)	XRT_BORDER_NONE	CSG

凡例領域の枠線のスタイルを指定します。有効なスタイルは XRT_BORDER_NONE、XRT_BORDER_PLAIN 、 XRT_BORDER_3D_IN 、 XRT_BORDER_3D_OUT 、 XRT_BORDER_ETCHED_IN 、 XRT_BORDER_ETCHED_OUT 、 XRT_BORDER_FRAME_IN 、 XRT_BORDER_FRAME_OUT、XRT_BORDER_BEVEL、XRT_BORDER_SHADOW です。

XtNxrtLegendBorderWidth	Dimension	2	CSG

凡例領域の枠線の幅を pixel で指定します。範囲は 0 から 20 です。

XtNxrtLegendFont	XmFontList	“fixed”	CSG

凡例に使用されるフォントを指定します。

XtNxrtLegendForegroundColor　char*　NULL　CSG

凡例領域の前景色を指定します。NULL であればウィンドウの前景色が使用されます。ストリングは色名か#で始まる RGB(赤、緑、青)構成の二桁の 16 進数です。色の指定についてはセクション 2.8 をご覧ください。

モノクロのサーバでは、白以外の色を指定しますと黒に表示されます。

XtNxrtLegendImage　XrtImage　NULL　CSG

凡例の領域の背景に表示するイメージのファイル名を指定します。

XtNxrtLegendImageLayout　XrtImageLayout　XRT_IMAGELAYOUT_CENTERED　CSG

背景のイメージの位置とサイズを指定します。有効な値は、XRT_IMAGELAYOUT_CENTERED か XRT_IMAGE_TILED です。

XtNxrtLegendOrientation　XrtAlign(enum)　XRT_ALIGN_VERTICAL　CSG

凡例の向きを指定します。有効な値は XRT_ALIGN_HORIZONTAL と XRT_ALIGN_VERTICAL です。

XtNxrtLegendReversed　Boolean　False　CSG

凡例に表示されるセットのエントリの順序が逆になるように指定します。

XtNxrtLegendShow　Boolean　True　CSG

凡例を表示するかしないかを指定します。

XtNxrtLegendTitle　char *　NULL　CSG

凡例のタイトルを指定します。凡例のフォントを使用した一行のタイトルを凡例の上中央に表示できます。

XtNxrtLegendHeight　Dimension　dynamic　G
XtNxrtLegendWidth　Dimension　dynamic　G

凡例領域の幅と高さ(pixel)を持っています。

XtNxrtLegendX　Position　**アンカーに依存**　CSG
XtNxrtLegendY　Position　**アンカーに依存**　CSG

凡例領域の左上角の(X,Y)座標を指定します。これらのリソースを設定すると、副作用で対応する UseDefault リソースは False になります。

XtNxrtLegendXUseDefault　Boolean　True　CSG
XtNxrtLegendYUseDefault　Boolean　True　CSG

True の場合、XtNxrtLegendX と XtNxrtLegendY の値は XRT/graph によって決定されます。XtNxrtLegendX と XtNxrtLegendY の値が設定されている時は、XRT/graph がこれらを False にします。XtNxrtLegendX と XtNxrtLegendY の値が提供されていないと、False にセットすることはできません。

XtNxrtMapCallback XtCallbackList NULL CSG

ユーザ・イベントのグラフ座標をアプリケーションに通知するために呼ばれるコールバックのリストを指定します。Map コールバックは、*XrtMap()*を使用する *Map()*アクション [1] によって開始されます。XrtMapCallbackStruct のポインタが各コールバックに渡されます。*XtAddCallback()*を用いてリストにコールバックを追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget           graph,
  caddr_t          client_data,
  XrtMapCallbackStruct      *call_data
  )
```

XrtMapCallbackStruct は次のように定義されます。

```
typedef struct {
  int          reason;  /* Read-only */
  XEvent       *event;  /* Read-only */
  String       *params; /* Read-only */
  Cardinal     nparams; /* Read-only */
  XrtRegion        region;  /* Read-only */
  XrtMapResult     map;/* Read-only */
} XrtMapCallbackStruct;
```

reason は、常に XRT_REASON_MAP です。*region* はイベントが起きたウィジェットの領域、*map* は、*XrtMap()*によって計算されたグラフ座標を持ちます。

XtNxrtMarkerDataset int XRT_DATASET1 CSG

複合グラフでどちらのデータセットにマーカを配置するか指定します。XtNxrtMarkerDataset を XRT_DATASET2 に変更し、XtNxrtXMarkerPoint、XtNxrtXMarkerSet、XtNxrt[XY]Marker を用いて二番目のデータセットに配置するようにできます。

XtNxrtMarkerDataStyle XrtDataStyle* **黒の破線** CSG

X と Y マーカの XrtDataStyle を指定します。XrtDataStyle の線のスタイル、色、幅が使用されます。指デフォルトでは、XRT/graph は黒、1 pixel 幅の破線をマーカに使用します。

このリソースは、XtNxrt[XY]MarkerDataStyle が明示的にセットされていないマーカにのみ使用されます。

XtNxrtMarkerDataStyleUseDefault Boolean True CSG

True の場合、デフォルトのマーカのデータ・スタイルが使用されます。データ・スタイルが設定されている場合は、XRT/graph はこれを False にします。

XtNxrtModifyCallback XtCallbackList NULL CSG

ユーザ・インタラクション(スケーリング、ズーム、移動、回転)の開始と終了をアプリケーションに通知するために呼ばれるコールバックのリストを指定します。Modify コールバックは、*ModifyStart()*と *ModifyEnd()*アクションによって開始されます。

[1] このアクションには、デフォルトのトランスレーションはありません。

XrtModifyCallbackStruct のポインタが各コールバックに渡されます。*XtAddCallback()* を用いてリストにコールバックを追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget           graph,
  caddr_t          client_data,
  XrtModifyCallbackStruct   *call_data
)
```

XrtModifyCallbackStruct は次のように定義されます。

```
typedef struct {
  int     reason;  /* Read-only */
  XEvent  *event;  /* Read-only */
  String  *params; /* Read-only */
  Cardinalnparams; /* Read-only */
  Boolean doit;
} XrtModifyCallbackStruct;
```

reason は、XRT_REASON_MODIFY_START または XRT_REASON_MODIFY_END のどちらかです。*doit* は、(*reason* が XRT_REASON_MODIFY_START の時)、インタラクションを許可するかどうかを決定するフラグです。全てのインタラクションを不可にするには、*reason* が XRT_REASON_MODIFY_START の時に、コールバック手続きで *doit* を False にセットします。

XtNxrtOtherDataStyle　XrtDataStyle*　**サーバに依存**　CSG

円グラフで"その他"のスライスを表示する時に使用される XrtDataStyle を指定します。XrtDataStyle の色と塗りつぶしパターンが使用されます。デフォルトで、XRT/graph は既にデータ表示に使用されているスタイルとは異なったスタイルをこのスライスに指定します。(XtNxrtDataStylesUseDefault が True の場合)

XtNxrtOtherDataStyleUseDefault　Boolean　True　CSG

True であれは、"その他"のスライスのスタイルは XRT/graph によって決定されます。XtNxrtDataSytlesUseDefault が True の場合、必ず異なったスタイルが選択されます。この機能は、コンパイル時にサーバがカラーかモノクロか不明で、サーバによって異なる設定を避ける場合に便利です。"その他"のスライスのスタイルが設定されていると、XRT/graph はこれを False にします。

XtNxrtOtherLabel　char*　"Other"　CSG

円グラフの"その他"のスライスのラベルを指定します。デフォルトのラベルは、"Other"です。無効な値(NULL や 0 の長さのストリング等)をセットしようとすると、XRT/graph は、"Other"というストリングを使用します。

XtNxrtPickCallback　XtCallbackList　NULL　CSG

ユーザ・イベントの pixel 座標に一番近いデータ値をアプリケーションに通知するために呼ばれるコールバックのリストを指定します。Pick コールバックは、*XrtPick()*を使用する *Pick()* アクション[1] によって開始されます。XrtPickCallbackStruct のポインタが各コールバックに渡されます。*XtAddCallback()*を用いてリストにコールバックを追加してください。

[1] このアクションには、デフォルトのトランスレーションはありません。

コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget           graph,
  caddr_t          client_data,
  XrtPickCallbackStruct *call_data
  )
```

XrtPickCallbackStruct は次のように定義されます。

```
typedef struct {
int           reason;  /* Read-only */
XEvent        *event;  /* Read-only */
String        *params; /* Read-only */
Cardinal      nparams; /* Read-only */
XrtRegion     region;  /* Read-only */
XrtPickResult      pick;    /* Read-only */
} XrtPickCallbackStruct;
```

reason は、常に XRT_REASON_PICK です。*region* は、イベントが起きたウィジェットの領域です。*pick* は、*XrtPick()*によって計算されたグラフのデータ値を持ちます。

XtNxrtPieMinSlices　int　1　CSG

円グラフに表示する最少スライス数("その他"のスライスも含む)を指定します。

XtNxrtPieOrder　XrtPieOrder(enum)　XRT_PIEORDER_DATA_ORDER　CSG

円グラフのスライスが表示される順序を指定します。XRT_PIEORDER_DESCENDING、XRT_PIEORDER_ASCENDING または XRT_PIEORDER_DATA_ORDER のどれかです。

XtNxrtPieStartAngle　float　135.0　CSG

円グラフの開始ポイントを指定します。デフォルトは、135 度です。

float 型のリソースをセットする時は、*XrtFloatToArgVal()*を使用してください。

XtNxrtPieThresholdMethod　XrtPieThresholdMethod　XRT_PIE_SLICE_CUTOFF　CSG

どの値を"その他"のスライスにグループ化するかを決定するしきい値法を指定します。XRT_PIE_SLICE_CUTOFF か XRT_PIE_PERCENTILE のどちらかです。

XRT_PIE_SLICE_CUTOFF であれば、XtNxrtPieThresholdValue の値より小さい割合の値は全て、"その他"のスライスにグループ化されます。

XRT_PIE_PERCENTILE であれば、"その他"のスライスが XtNxrtPieThresholdValue と同じかそれ以下になるのに必要なだけの値がグループ化されます。

例えば、"その他"のスライスを全体の 5%以下にする時は、XtNxrtPieThresholdMethod を XRT_PIE_PERCENTILE にセットし、XtNxrtPieThresholdValue を 5.0 に設定します。

但し、XtNxrtPieMinSlices は XtNxrtPieThresholdMethod より優先されます。

XtNxrtPieThresholdValue　float　0.0　CSG

XtNxrtPieThresholdMethod に使用される値を指定します。範囲は 0.0 から 100.0 です。"その他"のスライスを作成しない場合はこれを 0.0 に設定します。

float 型のリソースを設定する時は *XrtFloatToArgVal()*を使用してください。

XtNxrtPointLabels	char**	NULL	CSG
XtNxrtPointLabels2	char**	NULL	CSG

XtNxrtAnnotationMethod が XRT_ANNO_POINT_LABELS の時に、各ポイントのラベルに使用されるテキスト・ストリングのリストを定義します。XRT/graph は、下記を実行します。

- 各ポイントの X 値にラベルを表示する
- XtNxrtXAxisLabelFormat 、 XtNxrtXNumMethod 、 XtNxrtXNum 、 XtNxrtXPrecision を無視する

XtNxrtXAnnotationMethod が XRT_ANNO_POINT_LABELS に設定されていないと、このリソースは意味を持ちません。ポイント・ラベルを全てクリアするには、NULL にしてください。

XtNxrtPointLabels2 は、Y2 軸が primary(第一の)軸とされている時にのみ使用されます。(XtNxrtData が NULL)

XtNxrtPolarAxisAllowNegatives	Boolean	False	CSG

True にセットされると、負の値をポーラー・グラフに使用できるようになります。

XtNxrtPolarHalfRange	Boolean	False	CSG

ポーラー・グラフに使用される X 軸の範囲を指定します。False に設定されている場合は、デフォルトの X 軸の範囲は、0 から 360 度になります。True の場合は、軸の範囲は-180 度から 180 度になります。範囲は、XtNxrtAngleUnit で指定されたユニットを使用します。

XtNxrtRepaint	Boolean	True	CSG

True の場合、ウィジェットへの変更は直ちにアップデートされます。変更をバッチ処理する場合は、このリソースを False にしてから変更を加え、その後、リソースを True に戻します。

XtNxrtResizeCallback	XtCallbackList	NULL	CSG

XRT/graph がグラフをリサイズした時に、アプリケーションに通知するために呼ばれるコールバックのリストを指定します。XrtCallbackStruct のポインタが各コールバックに渡されます。*XtAddCallback()*手続きを使用してリストにコールバックを加えます。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget       graph,
  caddr_t      client_data,
  XrtCallbackStruct*call_data
  )
```

XrtCallbackStruct は次のように定義されます。

```
typedef struct {
  int      reason; /* Read-only */
  XEvent   *event; /* Read-only */
  Window   window; /* Read-only */
  Region   region; /* Read-only */
  int      change; /* Read-only */
} XrtCallbackStruct;
```

reason は常に XmCR_RESIZE です。*event*、*region*、*change* は常に NULL です。

XtNxrtRotateCallback	XtCallbackList	NULL	CSG

XRT/graph が 3D 棒または円グラフを回転しようとする時に、アプリケーションに通知するために呼ばれるコールバックのリストを指定します。

Rotate コールバックは、*RotateUpdate()* アクションによって開始されます。XrtRotateCallbackStruct のポインタが各コールバックに渡されます。*XtAddCallback()* 手続きを使用してリストにコールバックを加えます。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget           graph,
  caddr_t          client_data,
  XrtRotateCallbackStruct   *call_data
  )
```

XrtRotateCallbackStruct は次のように定義されます。

```
typedef struct {
  int     reason; /* Read-only */
  XEvent  *event; /* Read-only */
  String  *params;/* Read-only */
  Cardinalnparams;/* Read-only */
  int     rotation;
  int     inclination;
  Boolean doit;
} XrtRotateCallbackStruct;
```

reason は常に XRT_REASON_ROTATE です。*rotation* と *inclination* は、XtNxrtGraphRotation と XtNxrtGraphInclination に伝えられた値(これらの値は、コールバックで変更することができます)。*doit* は、インタラクションを許可するかどうか決定するフラグです。回転を不可にするには、コールバック手続きで *doit* を False にセットしてください。

XtNxrtSetLabels	char**	NULL	CSG
XtNxrtSetLabels2	char**	NULL	CSG

セット値に対応するストリングを指定します。セット・ラベルはグラフの凡例に表示されます。全てのセット・ラベルをクリアするには、XtNxrtSetLabels を NULL にします。

XtNxrtSetLabels2 は、複合グラフで二番目のグラフのセット・ラベルにアクセスします。

XtNxrtTimeBase	time_t	dynamic	CSG

1970 年 1 月 1 日 GMT からの時間データのオフセットを指定します。デフォルト値は、使用しているタイムゾーンの 1970 年 1 月 1 日です。タイムゾーンが EST であれば、このリソースのデフォルトは、(EST は Greenwich より 5 時間西なので)18000 になります。このリソースは、XtNxrtXAnnotationMethod が XRT_XMETHOD_TIME_LABELS の時使用されます。

time_t は<*time.h*>で定義され、通常 *long* に typedef されています。ANSI C の *mktime()*手続きは、このリソースに適する値を計算する時に便利です。

XtNxrtTimeFormat	char*	dynamic	CSG

軸の時間ラベルのフォーマット・ストリングを指定します。このリソースは、XtNxrtXAnnotationMethod が XRT_ANNO_TIME_LABELS の時にのみ使用されます。XRT/graph は、このリソースで指定されたストリングを、使用しているシステムの POSIX と ANSI C の *strftime()*と共に使用します。尚、フォーマットのオプションについては、システム関連の資料をご覧ください。

以下はANSIのフォーマット・オプションです。

%% パーセント(%)文字と同じ
%a locale の省略された週の名前
%A locale の週の名前
%b locale の省略された月の名前
%B locale の月の名前
%c locale の日付と時間
%d 一ヶ月の日(01-31)
%H 時間(00-23)
%I 時間(01-12)
%j 一年の日(001-366)
%m 月(01-12)
%M 分(00-59)
%p locale の AM か PM 相当
%S 秒(00-61)、閏秒を含む
%U 一年の週(00-53)、日曜日は週1の始めの日
%w 週の日(0-6)、日曜日=0
%W 一年の週(00-53)、月曜日は週1の始めの日
%x locale の日付
%X locale の時間
%y 一世紀の年(00-99)
%Y ccyy 年(例:1986)
%Z タイムゾーン名、ゾーンがない時は文字なし

NULL の場合、XRT/graph は、データの範囲と XtNxrtTimeUnit の値に基づいてデフォルトのフォーマットを生成します。

XtNxrtTimeFormatUseDefault Boolean True CSG

True の場合、XRT/graph はデフォルトの時間フォーマットをダイナミックに計算します。フォーマットが設定されている時は、XRT/graph はこれを False にします。

XtNxrtTimeUnit XrtTimeUnit(enum) XRT_TMUNIT_SECONDS CSG

XtNxrtXAnnotationMethod が XRT_ANNO_TIME_LABELS の時、データの X 値を解釈するユニットを指定します。有効な値は、XRT_TMUNIT_SECONDS、XRT_TMUNIT_MINUTES、XRT_TMUNIT_HOURS、XRT_TMUNIT_DAYS、XRT_TMUNIT_WEEKS、XRT_TMUNIT_MONTHS、XRT_TMUNIT_YEARS です。

例えば、範囲が 0 から 4 までの X 値を持つデータがグラフに接続されており、XtNxrtTimeUnit が XRT_TMUNIT_WEEKS にセットされている場合、XRT/graph は XtNxrtTimeBase で指定された時間から開始して、4週間の注釈を X 軸に付けます。

XtNxrtTransformCallback XtCallbackList NULL CSG

XRT/graph がグラフを移動、ズーム、スケーリングしようとする時に、アプリケーションに通知するために呼ばれるコールバックのリストを指定します。Transform コールバックは、*TranslateUpdate()* 、 *ZoomEnd()* 、 *ViewReset()* アクションによって開始されます。XrtTransformCallbackStruct のポインタが各コールバックに渡されます。*XtAddCallback()*手続きを使用してリストにコールバックを加えます。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget           graph,
  caddr_t          client_data,
  XrtTransformCallbackStruct    *call_data
)
```

XrtTransformCallbackStruct は次のように定義されます。

```
typedef struct {
  int     reason;  /* Read-only */
  XEvent  *event;  /* Read-only */
  String  *params; /* Read-only */
  Cardinalnparams; /* Read-only */
  Boolean reset;   /* Read-only */
  short   left_margin;
  short   right_margin;
  short   top_margin;
  short   bottom_margin;
  Boolean doit;
} XrtTransformCallbackStruct;
```

reason は常に XRT_REASON_TRANSFORM です。*reset* は、全てのスケーリング、移動、ズームをデフォルトにリセットにするかどうか通知します。*left_margin*、*right_margin*、*top_margin*、*bottom_margin* は、Margin リソースに伝えられた値です(これらの値は、コールバックで変更することができます)。*doit* は、インタラクションを許可するかどうか決定するフラグです。スケーリングや移動を不可にするには、コールバック手続きで *doit* を False にセットしてください。

XtNxrtType XrtType(enum) XRT_TYPE_PLOT CSG
XtNxrtType2 XrtType(enum) XRT_TYPE_BAR CSG

グラフ・タイプを決定します。XtNxrtType2 は、複合グラフの二番めのグラフのタイプを決定します。有効な値は、XRT_TYPE_PLOT 、 XRT_TYPE_AREA 、 XRT_TYPE_BAR 、XRT_TYPE_STACKING_BAR、XRT_TYPE_PIE、XRT_TYPE_HILO、XRT_TYPE_HILO_OPEN_CLOSE、XRT_TYPE_CANDLE、XRT_TYPE_POLAR、XRT_TYPE_RADAR、XRT_TYPE_FILLED_RADAR です。

XtNxrtXAnnoPlacement XrtAnnoPlacement(enum) XRT_ANNO_AUTO CSG
XtNxrtYAnnoPlacement XrtAnnoPlacement(enum) XRT_ANNO_AUTO CSG
XtNxrtY2AnnoPlacement XrtAnnoPlacement(enum) XRT_ANNO_AUTO CSG

軸の注釈とタイトルの位置を指定します。XRT_ANNO_AUTO であれば、原点(棒、HiLo、HiLoOpenClose、キャンドルでは軸の最大か最小で原点に近い方)に配置されます。これらのリソースは、ポーラー、レーダー、filled レーダーでは無視されます。XtNxrtYAnnoPlacement は二本軸のグラフでは無視されます。XtNxrtY2AnnoPlacement は、Y2 軸が primary(一番目の)Y 軸でなければ無視されます。

その他の値は、それぞれ注釈とタイトルを、XRT_ANNO_ORIGIN は原点の近くに、XRT_ANNO_MIN は軸の最小値に、XRT_ANNO_MAX は軸の最大値に配置します。

XtNxrtXAnnotationMethod	XrtAnnoMethod(enum)	XRT_ANNO_VALUES	CSG
XtNxrtYAnnotationMethod	XrtAnnoMethod(enum)	XRT_ANNO_VALUES	CSG
XtNxrtY2AnnotationMethod	XrtAnnoMethod(enum)	XRT_ANNO_VALUES	CSG

軸の注釈法を指定します。XRT_ANNO_VALUES、XRT_ANNO_POINT_LABELS、XRT_ANNO_VALUE_LABELS、XRT_ANNO_TIME_LABELS は、X 軸に有効な値です。Y と Y2 軸に有効な値は、XRT_ANNO_VALUES と XRT_ANNO_VALUE_LABELS です。無効な値にセットしようとすると、XRT_ANNO_VALUES が使用されます。

- XRT_ANNO_VALUES の場合は、グラフのX値または Y 値と XtNxrt[XYY2]NumMethod、XtNxrt[XYY2]Num、XtNxrt[XYY2]Tick、XtNxrt[XYY2]Precision の値を使用して、軸は適する間隔で注釈される。XtNxrt[XYY2]AxisLabelFormat のフォーマットが使用される。
- XRT_ANNO_VALUE_LABELS の場合は、XtNxrt[XYY2]Labels の値が注釈に使用される。レーダーと filled レーダーの X 軸には、この方法は使用できない。
- XRT_ANNO_POINT_LABELS の場合は、XtNxrtPointLabels(または XtNxrtPointLabels2 の値)が X 軸の注釈に使用される。General(一般)データのグラフでは、ポイント・ラベルは意味を持たない。
- XRT_ANNO_TIME_LABELS の場合は、X 軸に時間ラベルが使用される。ポーラー、レーダーと filled レーダーの X 軸には、この方法は使用できない。

XtNxrtXAnnotationRotation	XrtRotate(enum)	XRT_ROTATE_NONE	CSG
XtNxrtYAnnotationRotation	XrtRotate(enum)	XRT_ROTATE_NONE	CSG
XtNxrtY2AnnotationRotation	XrtRotate(enum)	XRT_ROTATE_NONE	CSG

軸の注釈テキストを回転させる場合にこのリソースを使用します。XRT_ROTATE_90 または XRT_ROTATE_270 にセットされている時、注釈は時計と反対方向に 90 度または 270 度に回転します。

回転されたテキストを使用すると、リアルタイムの速度が落ちますので、秒単位のアップデートが必要なグラフには使用しないでください。

XtNxrtXAnnotationMethod は、ポーラー、レーダー、filled レーダーでは無視されます。

XtNxrtXAxisDataStyle	XrtDataStyle *	**幅 1 の黒い実線**	CSG
XtNxrtYAxisDataStyle	XrtDataStyle *	**幅 1 の黒い実線**	CSG
XtNxrtY2AxisDataStyle	XrtDataStyle *	**幅 1 の黒い実線**	CSG

軸を表示する時に使用される XrtDataStyle を指定します。XrtDataStyle 構造体の、ポイント・サイズ(目盛の長さは、これを基準とする)、線のスタイル、色、幅等のエレメントが使用されます。このリソースが指定されていない場合は、1pixel 幅の実線をグラフの前景色(デフォルトでは通常、黒)で描画します。

これらのリソースをセットすると、副作用で対応する UseDefault リソースが False にセットされます。

XtNxrtXAxisDataStyleUseDefault	Boolean	True	CSG
XtNxrtYAxisDataStyleUseDefault	Boolean	True	CSG
XtNxrtY2AxisDataStyleUseDefault	Boolean	True	CSG

True の場合、デフォルトの軸のスタイルが使用されます。スタイルが明示的にセットされている場合は、この Boolean は False にセットされます。

XtNxrtXAxisLabelFormat	char*	NULL	CSG
XtNxrtYAxisLabelFormat	char*	NULL	CSG
XtNxrtY2AxisLabelFormat	char*	NULL	CSG

軸の注釈法が XRT_ANNO_VALUES の時、どのように軸の注釈をフォーマットするか指定します。NULL にセットされている場合、注釈はフォーマットされません。軸のラベルのフォーマットに関しては、4.4 にも解説がありますので参照してください。有効なフォーマット・コードは次の通りです。

#0?	**数値が入る場所を指定。 . (ピリオド)と一緒に使用して表示桁数を指定。0 は値に指定した桁が存在しない場合に 0 を挿入。?は 0 と同様の働きをし、小数点や分数を行揃えするために、指定した桁が存在しない場合にスペースを挿入。**
. (ピリオド)	**小数点を挿入**
, (カンマ)	1000 **毎の桁区切り。値の最後では、その値を** 1000 **で割る。**
$ - + / ( )	**指定場所に、これらの文字を挿入。**
\<char>	**指定場所に、**<char>**の文字を挿入。**
_<char>	**指定場所に、**<char>**の代りにスペースを挿入。**
%	**数値を** 100 **倍し、%を加える。**
[multiplier <n>]	**数値を**<n>**倍する。**
E+ e+ E- e-	**指数表記する。この記号の後には、指数の桁数を表す** placeholder(**数値が入る場所**)**を付ける。**
*<char>	**注釈に適当なスペースを挿入。**
[hide] [show]	**次に続く値/フォーマットを非表示/表示する。**
: (セミコロン)	**フォーマットのグループを区切る。**
[>val] [<val] [=val] [< >val] [>=val] [<=val]	**条件を指定して、フォーマット用のグループを定義する。値が条件に合った場合は、それに続くコードが適用される。**

XtNxrtXAxisLogarithmic	Boolean	False	CSG
XtNxrtYAxisLogarithmic	Boolean	False	CSG
XtNxrtY2AxisLogarithmic	Boolean	False	CSG

True の場合、XRT/graph はデータを対数(logarithmic)軸のグラフに表示します。0 以下の値は全て hole(欠けているデータ値)として表示されます。XtNxrt[XYY2]Tick、XtNxrt[XYY2]Num、XtNxrt[XYY2]Precision のリソースはセットできません。対数軸は、0 より大きい値のみをグラフ化するので、XtNxrt[XYY2]AxisMin、XtNxrt[XYY2]AxisMax、XtNxrt[XYY2]Min、XtNxrt[XYY2]Max、XtNxrt[XYY2]Origin の値を 0 以下にセットすると、対応する UseDefault のリソースが True になります。

XtNxrtXAxisLogarithmic は、棒、積み上げ棒、ポーラー、レーダー、filled レーダー・グラフや、X 軸の注釈法が XRT_ANNO_TIME_LABELS であると無視されます。円グラフでは、対数に関するリソースは無視されます。

XtNxrtXAxisMax	float	dynamic	CSG
XtNxrtYAxisMax	float	dynamic	CSG
XtNxrtY2AxisMax	float	dynamic	CSG

軸の最大値を指定します。グラフ・データは、XtNxrt[XYY2]AxisMax と XtNxrt[XYY2]Max の値以下になるようにカットされます。このリソースを設定すると、副作用で、対応する UseDefault リソースは、False にセットされます。

値が設定されていなければ、このリソースの値は XtNxrt[XYY2]Max、XtNxrt[XY]Origin、XtNxrt[XYY2]Num、XtNxrt[XYY2]NumMethod の値から派生し、XtNxrt[XYY2]Num の整数倍になります。

(注意) 対応する UseDefault リソースが False の時に、XtNxrt[XYY2]AxisMax を無効にするような軸の変更が加えられると、XRT/graph は UseDefault を True にセットし、軸の範囲は再計算されます。

XtNxrt[XYY2]AxisMax は、円、ポーラー、レーダー、filled レーダー・グラフでは無視されます。一番めと二番めの Y 軸の関係が設定されている場合は、XtNxrtY2AxisMax のデフォルトは、XtNxrtYAxisMax から計算されます。

float 型のリソースを設定する時は、*XrtFloatToArgVal()*を使用してください。

XtNxrtXAxisMaxUseDefault	Boolean	True	CSG
XtNxrtYAxisMaxUseDefault	Boolean	True	CSG
XtNxrtY2AxisMaxUseDefault	Boolean	True	CSG

True であれば、XRT/graph が軸の最大値を決定します。XtNxrt[XYY2]AxisMax の値が設定されていると、このリソースは False にセットされます。XtNxrt[XYY2]AxisMax の値が与えられていないと、このリソースは False にセットできません。

(注意) このリソースが False の時に、XtNxrt[XYY2]AxisMax を無効にするような軸の変更が加えられると、XRT/graph は UseDefault を True にセットし、軸の範囲は再計算されます。

XtNxrtXAxisMin	float	dynamic	CSG
XtNxrtYAxisMin	float	dynamic	CSG
XtNxrtY2AxisMin	float	dynamic	CSG

軸の最小値を指定します。グラフ・データは、XtNxrt[XYY2]AxisMin と XtNxrt[XYY2]Min の値以上になるようにカットされます。このリソースを設定すると、副作用で、対応する UseDefault リソースは、False にセットされます。

値が設定されていなければ、このリソースの値は XtNxrt[XYY2]Min、XtNxrt[XY]Origin、XtNxrt[XYY2]Num、XtNxrt[XYY2]NumMethod の値から派生し、XtNxrt[XYY2]Num の整数倍になります。

(注意) 対応する UseDefault リソースが False の時に、XtNxrt[XYY2]AxisMin を無効にするような軸の変更が加えられると、XRT/graph は UseDefault を True にセットし、軸の範囲は再計算されます。

XtNxrt[XYY2]AxisMin は、円、ポーラー、レーダー、filled レーダー・グラフでは無視されます。一番めと二番めの Y 軸の関係が設定されている場合は、XtNxrtY2AxisMin のデフォルトは、XtNxrtYAxisMin から計算されます。

float 型のリソースを設定する時は、*XrtFloatToArgVal()*を使用してください。

XtNxrtXAxisMinUseDefault	Boolean	True	CSG
XtNxrtYAxisMinUseDefault	Boolean	True	CSG
XtNxrtY2AxisMinUseDefault	Boolean	True	CSG

`True` であれば、XRT/graph が軸の最小値を決定します。`XtNxrt[XYY2]AxisMin` の値が設定されていると、このリソースは `False` にセットされます。`XtNxrt[XYY2]AxisMin` の値が与えられていないと、このリソースは `False` にセットできません。

(注意) このリソースが `False` の時に、`XtNxrt[XYY2]AxisMin` を無効にするような軸の変更が加えられると、XRT/graph は `UseDefault` を `True` にセットし、軸の範囲は再計算されます。

XtNxrtXAxisReversed	Boolean	False	CSG
XtNxrtYAxisReversed	Boolean	False	CSG
XtNxrtY2AxisReversed	Boolean	False	CSG

`True` の場合、XRT/graph は軸と注釈を反対の方向に表示します。この方法は、X 軸を右から左の方向に増加させたり、Y 値を下方向に増加させる場合に使用できます。

`XtNxrtY2AxisReversed` は、`XtNxrtYAxisMult` が非 0 の時は無視されます。二番目の Y 軸は、正の乗数が指定されると一番目の Y 軸と同じ方向に、負の乗数が指定されると反対方向になります。

XtNxrtXAxisShow	Boolean	True	CSG
XtNxrtYAxisShow	Boolean	True	CSG
XtNxrtY2AxisShow	Boolean	False	CSG

軸の表示の有無を指定します。

全ての Y2 軸のリソースは、二番目のデータセットが(`XtNxrtData2` を使用して)接続されていなかったり、`XtNxrtYAxisMult` が非 0 でないと無視されます。

XtNxrtXGrid	float	0.0	CSG
XtNxrtYGrid	float	0.0	CSG
XtNxrtY2Grid	float	0.0	CSG

原点から開始して、グリッド線の増分(increment)を指定します。Y2 軸が primary(第一の)Y 軸でない場合は、`XtNxrtY2Grid` は無視されます。これらのリソースを設定すると、副作用で、対応する `UseDefault` リソースは `False` にセットされます。

この値が 0.0 の場合は、グリッド線は表示されません。

`XtNxrt[XYY2]GridUseDefault` が `True` にセットされると、デフォルトのグリッド線の増分は `XtNxrt[XYY2]Num` の値になり、グリッド線は各軸ラベル上に表示されます。

float 型のリソースを設定する時は、*XrtFloatToArgVal()*を使用してください。

XtNxrtXGridUseDefault	Boolean	False	CSG
XtNxrtYGridUseDefault	Boolean	False	CSG
XtNxrtY2GridUseDefault	Boolean	False	CSG

`True` の場合、XRT/graph がグリッド線の増分(increment)を決定します。`XtNxrt[XYY2]Grid` の値が設定されている場合は、これらのリソースは `False` にセットされます。`XtNxrt[XYY2]Grid` の値が設定されていないと、これらのリソースは `False` にセットできません。

`XtNxrt[XYY2]GridUseDefault` が `True` にセットされると、デフォルトのグリッド線の増分は `XtNxrt[XYY2]Num` の値になり、グリッド線は各軸ラベル上に表示されます。

XtNxrtXGridDataStyle	XrtDataStyle*	**点線**	CSG
XtNxrtYGridDataStyle	XrtDataStyle*	**点線**	CSG
XtNxrtY2GridDataStyle	XrtDataStyle*	**点線**	CSG

X と Y のグリッド線を表示する時の `XrtDataStyle` を指定します。`XrtDataStyle` の線スタイル、色、幅が使用されます。指定されていない場合は、1 pixel 幅の点線でグラフの前景色を使用したグリッド線が表示されます。このリソースを設定すると副作用で、対応する `UseDefault` リソースは `False` にセットされます。Y2 軸が primary(第一の)Y 軸でない場合は、`XtNxrtY2GridDataStyle` は無視されます。

XtNxrtXGridDataStyleUseDefault	Boolean	True	CSG
XtNxrtYGridDataStyleUseDefault	Boolean	True	CSG
XtNxrtY2GridDataStyleUseDefault	Boolean	True	CSG

`True` の場合、デフォルトのグリッド線スタイルが使用されます。スタイルが指定されている場合は、このリソースは `False` にセットされます。

XtNxrtXLabels	XrtValueLabel**	NULL	CSG
XtNxrtYLabels	XrtValueLabel**	NULL	CSG
XtNxrtY2Labels	XrtValueLabel**	NULL	CSG

`XtNxrt[XYY2]AnnotationMethod` が `XRT_ANNO_VALUE_LABELS` の時に、X、Y、Y2 軸のラベルを提供します。このリソースの値は、`XrtValueLabel` 構造体のポインタの `NULL` 終端配列です。各構造体は、値とラベルのペアを持っています。

Value ラベルは、円グラフの注釈や、レーダー、filled レーダー・グラフの X 軸の注釈には使用できません。

*XtGetValues()*で使用される時、値-ラベルのペアが定義されていないと `NULL` ポインタが返されます。返された Value ラベルは、X 値または Y 値によってソートされます。

XtNxrtXMarker	float	0.0	CSG
XtNxrtYMarker	float	0.0	CSG

グラフ座標でマーカの位置を指定します。対応する `MarkerShow` リソースが `True` で、X マーカの場合は、`XtNxrtXMarkerMethod` が `XRT_MARKER_VALUE` にセットされていないとマーカは表示されません。複合グラフでマーカを配置する場合、これらのリソースを使用する前に `XtNxrtMarkerDataset` を設定してください。

`XtNxrt[XY]Marker` は、円、レーダー、filled レーダー・グラフでは無視され、`XtNxrtYMarker` はポーラー・グラフでは無視されます。

float 型のリソースを設定する時は *XrtFloatToArgVal()*を使用してください。

XtNxrtXMarkerDataStyle	XrtDataStyle*	dynamic	CSG
XtNxrtYMarkerDataStyle	XrtDataStyle*	dynamic	CSG

X または Y のマーカを表示する時の XrtDataStyle を指定します。XrtDataStyle の線スタイル、色、幅が使用されます。指定されていない場合は、XtNxrtMarkerDataStyle の値を適用したマーカ(デフォルトは黒い 1 pixel 幅の点線)が表示されます。XtNxrt[XY]MarkerDataStyle がセットされた時は、XtNxrtMarkerDataStyle の代りに、この設定が使用されます。

XtNxrtXMarkerDataStyleUseDefault	Boolean	True	CSG
XtNxrtYMarkerDataStyleUseDefault	Boolean	True	CSG

True の場合は、マーカはデフォルトのデータ・スタイルを使用します。データ・スタイルが設定されている場合は、このリソースは False にセットされます。

XtNxrtXMarkerMethod	XrtMarkerMethod	XRT_MARKER_SET_POINT	CSG

X マーカの位置決めにどのリソースを使用するか指定します。XRT_MARKER_POINT にセットされた場合は、XtNxrtXMarkerSet と XtNxrtXMarkerPoint が使用されます。XRT_MARKER_VALUE にセットされた場合は、XtNxrtXMarker が使用されます。

円、レーダー、filled レーダー・グラフは、X マーカの位置決めにセット・ポイントしか使用できません。

XtNxrtXMarkerPoint	int	0	CSG
XtNxrtXMarkerSet	int	0	CSG

X マーカが表示されるデータのセットとポイントのインデックスを指定します。円、レーダー、filled レーダー・グラフには、この方法でマーカが指定できます。マーカは、対応する MarkerShow が True で、XtNxrtXMarkerMethod が XRT_MARKER_SET_POINT ないと表示されません。

指定したポイントの Y 値が hole 値であれば、マーカは表示されません。複合グラフでマーカを表示する時は、これらのリソースを使用する前に XtNxrtMarkerDataset をセットしてください。

XtNxrtXMarkerShow	Boolean	False	CSG
XtNxrtYMarkerShow	Boolean	False	CSG

マーカの表示の有無を指定します。XtNxrt[XY]Marker、XtNxrtXMarkerSet、XtNxrtXMarkerPoint はマーカの位置、XtNxrtXMarkerMethod が X マーカの方法を指定します。

XtNxrtXMax	float	dynamic	CSG
XtNxrtYMax	float	dynamic	CSG
XtNxrtY2Max	float	dynamic	CSG

X、Y、Y2 データの最大値を指定します。この値より大きいデータ値は表示されません。これらのリソースを設定すると副作用で、対応する UseDefault リソースが False になります。

XtNxrt[XYY2]Max は円グラフでは無視されます。ポーラー、レーダー、filled レーダー・グラフでは、XtNxrtXMax の設定は意味を持たず、XtNxrtYMax は、データの最大 Y 値より小さくセットすることはできません。

一番目の Y 軸と二番目の Y 軸に関係がある場合は、XtNxrtY2Max のデフォルトは XtNxrtYMax から計算されます。

float 型のリソースを設定する場合は *XrtFloatToArgVal()*を使用してください。

XtNxrtXMaxUseDefault	Boolean	True	CSG
XtNxrtYMaxUseDefault	Boolean	True	CSG
XtNxrtY2MaxUseDefault	Boolean	True	CSG

True の場合、XtNxrt[XYY2]Max 値は XRT/graph によって決定されます。XtNxrt[XYY2]Max 値が与えられている場合は、XRT/graph はこのリソースを False にセットします。XtNxrt[XYY2]Max の値が提供されていないと、これらを False にすることはできません。

XtNxrtXMin	float	dynamic	CSG
XtNxrtYMin	float	dynamic	CSG
XtNxrtY2Min	float	dynamic	CSG

X、Y、Y2 データの最小値を指定します。この値より小さいデータ値は表示されません。これらのリソースを設定すると副作用で、対応する UseDefault リソースが False になります。

XtNxrt[XYY2]Min は円グラフでは無視されます。ポーラー、レーダー、filled レーダー・グラフでは、XtNxrtXMin の設定は意味を持たず、XtNxrtYMin は、データの最小 Y 値より大きくセットすることはできません。

一番目の Y 軸と二番目の Y 軸に関係がある場合は、XtNxrtY2Min のデフォルトは XtNxrtYMin から計算されます。

float 型のリソースを設定する場合は *XrtFloatToArgVal()*を使用してください。

XtNxrtXMinUseDefault	Boolean	True	CSG
XtNxrtYMinUseDefault	Boolean	True	CSG
XtNxrtY2MinUseDefault	Boolean	True	CSG

True の場合、XtNxrt[XYY2]Min 値は XRT/graph によって決定されます。XtNxrt[XYY2]Min に値が与えられている場合は、XRT/graph はこれらのリソースを False にセットします。XtNxrt[XYY2]Min の値が提供されていないとこれらを False にすることはできません。

XtNxrtXNum	float	dynamic	CSG
XtNxrtYNum	float	dynamic	CSG
XtNxrtY2Num	float	dynamic	CSG

軸の数値ラベルの増分(increment)を指定します。これらのリソースを設定すると、副作用で対応する `UseDefualt` リソースが `False` にセットされます。

円グラフや軸が対数(logarithmic)の場合は、`XtNxrt[XYY2]Num` は無視されます。

`XtNxrt[XYY2]Num` のデフォルト値は、対応する `XtNxrt[XYY2]Tick` が明示的に設定されていない場合は、データから派生します。設定されている場合は、数値ラベルの増分は、目盛の増分の二倍になります。

float 型のリソースを設定する場合は、*XrtFloatToArgVal()*を使用してください。

XtNxrtXNumUseDefault	Boolean	True	CSG
XtNxrtYNumUseDefault	Boolean	True	CSG
XtNxrtY2NumUseDefault	Boolean	True	CSG

`True` の場合、`XtNxrt[XYY2]Num` 値は XRT/graph によって決定されます。`XtNxrt[XYY2]Num` の値が与えられている場合は、XRT/graph はこれらを `False` にします。`XtNxrt[XYY2]Num` の値が提供されていないとこれらを `False` にすることはできません。

XtNxrtXNumMethod	XrtNumMethod(enum)	XRT_NUM_ROUND	CSG
XtNxrtYNumMethod	XrtNumMethod(enum)	XRT_NUM_ROUND	CSG
XtNxrtY2NumMethod	XrtNumMethod(enum)	XRT_NUM_ROUND	CSG

軸の注釈を計算する時に用いられる方法を指定します。これらのリソースは、`XtNxrt[XYY2]Num` リソースのデフォルト値をコントロールします。`XRT_NUM_PRECISION` であれば、軸の精度 (`XtNxrt[XYY2]Precision`) の現在値が使用されます。`XRT_NUM_ROUND` であれば、数値は四捨五入されます。どちらの場合も、軸の精度の現在値が数値のフォーマットに使用されます。

XtNxrtXOrigin	float	dynamic	CSG
XtNxrtYOrigin	float	dynamic	CSG
XtNxrtY2Origin	float	dynamic	CSG

軸の位置を指定します。例えば、X の原点を 5.0 に設定すると、Y 軸は X 軸を X=5.0 で交差します。このリソースを設定すると副作用で、対応する `UseDefault` リソースが `False` になります。

円グラフでは `XtNxrt[XYY2]Origin` は無視されます。Y2 軸が primary(第一の)Y 軸でなければ、`XtNxrtY2Origin` は無視されます。

float 型のリソースを設定する場合は、*XrtFloatToArgVal()*を使用してください。

XtNxrtXOriginBase	float	0.0	CSG

ポーラー、レーダー、filled レーダー・グラフで、X 軸の原点として使用する角度を指定します。この角度を 0 とし、全ての角度は、ここを基準にします。このリソースの値は、`XtNxrtAngleUnit` で指定されるユニットで解釈されます。

float 型のリソースを設定する場合は、*XrtFloatToArgVal()*を使用してください。

XtNxrtXOriginPlacement	XrtOriginPlacement(enum)	XRT_ORIGIN_AUTO	CSG
XtNxrtYOriginPlacement	XrtOriginPlacement(enum)	XRT_ORIGIN_AUTO	CSG
XtNxrtY2OriginPlacement	XrtOriginPlacement(enum)	XRT_ORIGIN_AUTO	CSG

原点の位置を指定します。XRT_ORIGIN_AUTO であると、XRT/graph は次のように原点を配置します。

- 線または面グラフ、バブル・チャート: 原点は、軸の最小値または、データ・セットが正と負の両方の値を持つ場合には、0 に配置される。
- 棒、HiLo、HiLoOpenClose、キャンドル・グラフ: 0 に配置される。

XRT_ORIGIN_ZERO は 0 に、XRT_ORIGIN_MIN は軸の最小に、XRT_ORIGIN_MAX は軸の最大に、それぞれ原点を配置します。

積み上げ棒グラフでは、原点は常に 0 に配置されます。その他の場所に設定しようとしても無視されます。これらのリソースは、円グラフでは無視されます。また、Y2 軸が primary(第一の)Y 軸になっていなければ、XtNxrtY2OriginPlacement は無視されます。

このリソースは、XtNxrt[XYY2]Origin で原点を明示的に設定した時は無視されます。

XtNxrtXOriginUseDefault	Boolean	True	CSG
XtNxrtYOriginUseDefault	Boolean	True	CSG
XtNxrtY2OriginUseDefault	Boolean	True	CSG

True の場合、原点の値は XRT/graph によって決定されます。XtNxrt[XYY2]Origin の値が与えられている時は、これらのリソースは False にセットされます。XtNxrt[XYY2]Origin の値が提供されていないとこれを False にすることはできません。

XtNxrtXPrecision	int	dynamic	CSG
XtNxrtYPrecision	int	dynamic	CSG
XtNxrtY2Precision	int	dynamic	CSG

このリソースは、軸の数値ラベルの小数点以下の桁数を指定します。0 以下なら数値は整数になります。XRT/graph は、軸の原点/最小/最大、数値ラベル/目盛りの増分(increment)を決定する時にもこの精度を使用します。例えば、XtNxrtXOrigin が 1.45、XtNxrtXPrecision が 0 に設定されていると、Y 軸は X=1 に表示されます。XtNxrtXPrecision が 2 に変更されると Y 軸は X=1.45 に表示されます。

精度が負の値は、10 の累乗を示します。例えば、XtNxrtYPrecision を-2 に設定すると、Y 軸の数値は 100 の倍数になります。

このリソースを設定すると、副作用で対応する UseDefault リソースが False になります。

XtNxrtXPrecisionUseDefault	Boolean	True	CSG
XtNxrtYPrecisionUseDefault	Boolean	True	CSG
XtNxrtY2PrecisionUseDefault	Boolean	True	CSG

True の場合、XRT/graph は XtNxrt[XYY2]Precision の値を自動的に計算します。XtNxrt[XYY2]Precision に値が与えられていると、これらのリソースは False にセットされます。XtNxrt[XYY2]Precision の値が提供されていないとこれらを False にすることはできません。

XtNxrtXTick	float	dynamic	CSG
XtNxrtYTick	float	dynamic	CSG
XtNxrtY2Tick	float	dynamic	CSG

軸の目盛(tick)の増分(increment)を指定します。これらのリソースを設定すると副作用で対応する `UseDefault` が `False` にセットされます。

デフォルトの `XtNxrt[XYY2]Tick` は、対応する `XtNxrt[XYY2]Num` が設定されていない限りデータから派生します。設定されていると、目盛りの増分は数値の増分の半分になります。

`XtNxrt[XYY2]Tick` は円グラフや軸が対数の時は無視されます。

X 軸の注釈法がポイント・ラベル、値(Value)ラベルか時間ラベルで、`XtNxrtXTickUseDefault` が `True` の場合、目盛りは各ラベル上に表示されます。`XtNxrtXTickUseDefault` が `False` であれば、指定された X の目盛りの増分が使用されます。増分が 0 の時は、目盛りは表示されません。

float 型のリソースを設定する場合は、*XrtFloatToArgVal()*を使用してください。

XtNxrtXTickUseDefault	Boolean	True	CSG
XtNxrtYTickUseDefault	Boolean	True	CSG
XtNxrtY2TickUseDefault	Boolean	True	CSG

`True` の場合、`XtNxrt[XYY2]Tick` の値は XRT/graph によって決定されます。`XtNxrt[XYY2]Tick` に値が与えられている場合は、これらのリソースは `False` にセットされます。`XtNxrt[XYY2]Tick` の値が提供されていないと、これらを `False` にすることはできません。

XtNxrtXTitle	char*	NULL	CSG
XtNxrtYTitle	char*	NULL	CSG
XtNxrtY2Title	char*	NULL	CSG

X、Y、Y2 軸のタイトルを指定します。円、ポーラー、レーダー、filled レーダー・グラフではこのリソースは無効です。

XtNxrtXTitleRotation	XrtRotate(enum)	XRT_ROTATE_NONE	CSG
XtNxrtYTitleRotation	XrtRotate(enum)	XRT_ROTATE_NONE	CSG
XtNxrtY2TitleRotation	XrtRotate(enum)	XRT_ROTATE_NONE	CSG

タイトル・テキストの回転にこのリソースを使用します。これは縦軸(`XtNxrtInvertOrientation` が `True` でない限り Y 軸か Y2 軸)にのみ適用されます。`XRT_ROTATE_NONE` にセットされている場合、タイトルは軸の上または下の中央に配置されます。`XRT_ROTATE_90` か `XRT_ROTATE_270` の時は、タイトルは 90 度か 270 度に時計と反対の方向に回転し、縦軸の右か左の中央に配置されます。

回転されたテキストはリアルタイムの速度を落としますので、秒単位のアップデートが必要なグラフにはなるべく使用しないでください。

XtNxrtYAnnotationAngle	float	dynamic	CSG
XtNxrtY2AnnotationAngle	float	dynamic	CSG

Y と Y2 軸の注釈を表示する角度を指定します。このリソースは、ポーラー、レーダー、filled レーダー・グラフに使用されます。デフォルトは、現在の X 原点値(XtNxrtXOrigin)です。XtNxrtY2AnnoationAngle は、Y2 軸が primary(第一の)Y 軸の時にのみ使用されます。

これらのリソースを設定すると、副作用で、対応する UseDefault リソースが False にセットされます。

float 型のリソースを設定する場合は、*XrtFloatToArgVal()*を使用してください。

XtNxrtYAnnotationAngleUseDefault	Boolean	True	CSG
XtNxrtY2AnnotationAngleUseDefault	Boolean	True	CSG

True の場合、XRT/graph が XtNxrt[YY2]AnnotationAngle の値を自動的に決定します。XtNxrt[YY2]AnnotationAngle の値が与えられてる場合は、これらのリソースは False にセットされます。XtNxrt[YY2]AnnotationAngle の値が提供されていないと、これらは False にはできません。

XtNxrtYAxis100Percent	Boolean	False	CSG
XtNxrtY2Axis100Percent	Boolean	False	CSG

積み上げグラフで、Y 軸と Y2 軸の注釈にパーセンテージを使用するかどうかどうか指定します。True にセットされた場合、データは、Y 値の合計に対する割合として解釈されます。False の場合は、Y 値はパーセンテージを使用して解釈されません。

これらのリソースは、グラフの種類が XRT_TYPE_STACKING_BAR、またはスタッキングできるグラフ(XtNxrtIsStacked または XtNxrtIsStacked2 が True)の場合にのみ使用されます。

XtNxrtYAxisConst	float	0.0	CSG

XtNxrtYAxisMult と共に一番目と二番目の Y 軸の関係を指定します。二本の軸に関係性がない場合は、XtNxrtYAxisMult を 0.0 に設定します。

例えば、一番目の Y 軸が摂氏で二番目の Y 軸が華氏の場合は、XtNxrtYAxisMult を 9.0/5.0 に、XtNxrtYAxisConst を 32.0 に設定します。

float 型のリソースを設定する時は、*XrtFloatToArgVal()*を使用してください。

XtNxrtYAxisMult	float	0.0	CSG

XtNxrtYAxisConst と共に一番目と二番目の Y 軸の関係を指定します。二本の軸に関係性がない場合は、XtNxrtYAxisMult を 0.0 に設定します。

例えば、一番目の Y 軸が摂氏で二番目の Y 軸が華氏の場合は、XtNxrtYAxisMult を 9.0/5.0 に、XtNxrtYAxisConst を 32.0 に設定します。

float 型のリソースを設定する時は、*XrtFloatToArgVal()*を使用してください。

XtNxrtZoomAxisCallback	XtCallbackList	NULL	CSG

XRT/graph が、軸の範囲を変更してグラフをズームしようとする時にアプリケーションに通知するために呼び出されるコールバックのリストを指定します。ZoomAxis コールバックは、*ZoomAxisEnd()*アクションによって開始されます。XrtZoomAxisCallbackStruct のポインタが各コールバックに渡されます。

*XtAddCallback()*を使用して、リストにコールバックを追加してください。コールバックは、次のように構成されます。

```
void
CallbackProcedure(
  Widget           graph,
  caddr_t          client_data,
  XrtZoomAxisCallbackStruct *call_data
)
```

XrtZoomAxisCallback は次のように定義されます。

```
typedef struct {
  int     reason;  // Read-only
  XEvent  *event;  // Read-only
  String  *params; // Read-only
  Cardinalnparams; // Read-only
  Boolean reset;   // Read-only
  float   new_xaxis_min;
  float   new_xaxis_max;
  float   new_yaxis_min;
  float   new_yaxis_max;
  float   new_y2axis_min;
  float   new_y2axis_max;
  Boolean doit;
} XrtZoomAxisCallbackStruct;
```

reason は常に XRT_REASON_ZOOM_AXIS です。*reset* は、全ての軸のズームが XRT/graph のデフォルトに戻るかどうかを通知します。*new_xaxis_min* と *new_y2axis_max* は、軸の最初と最大のリソースに伝えられた値(これらの値は、コールバックで変更することができます)。*doit* は、インタラクションを許可するかどうかを決定するフラグです。軸のズームを不可にするには、コールバック手続きで *doit* を False にセットします。

A.3 継承されるリソース

XmPrimitive リソース・セット			
Name	**Type**	**Default**	**Access**
XmNhighlightColor	Pixel	dynamic	CSG
XmNhighlightOnEnter	Boolean	False	CSG
XmNhighlightPixmap	Pixmap	dynamic	CSG
XmNnavigationType	XmNavigationType	XmNONE	CSG
XmNshadowThickness	Dimension	2	CSG
XmNtraversalOn	Boolean	True	CSG
XmNuserData	XtPointer	NULL	CSG

Core リソース・セット			
Name	**Type**	**Default**	**Access**
XmNdestroyCallback	XtCallbackList	NULL	C
XmNheight	Dimension	dynamic	CSG
XmNtranslations	XtTranslations	Appendix E 参照	CSG
XmNwidth	Dimension	dynamic	CSG
XmNx	Position	0	CSG
XmNy	Position	0	CSG

B

テキスト・オブジェクトの属性

TextArea オブジェクトの全ての属性のアルファベット順の一覧です。属性名の後は、データ型とデフォルト値が記載してあります。

インスタンスの生成: `hTextObject = XrtTextAreaCreate(graph)`

属性の方法: *XrtTextAreaGetValues(), XrtTextAreaSetValues()*

XRT_TEXT_ADJUST XrtAdjust(enum) XRT_ADJUST_CENTER

テキスト・オブジェクト内で、どのように複数行のテキストが行揃えされるかを指定します。有効な値は、XRT_ADJUST_LEFT、XRT_ADJUST_CENTER、XRT_ADJUST_RIGHT です。

XRT_TEXT_ANCHOR XrtAnchor(enum) XRT_ANCHOR_BEST

テキスト・オブジェクトを貼り付けポイントに対して、どのように配置するかを指定します。有効な値は、XRT_ANCHOR_NORTH、XRT_ANCHOR_SOUTH、XRT_ANCHOR_EAST、XRT_ANCHOR_WEST、XRT_ANCHOR_NORTHWEST 、 XRT_ANCHOR_NORTHEAST 、 XRT_ANCHOR_SOUTHWEST 、 XRT_ANCHOR_SOUTHEAST、XRT_ANCHOR_HOME、XRT_ANCHOR_BEST です。

XRT_TEXT_ATTACH_DATASET int XRT_DATASET1

テキスト・オブジェクトを貼り付ける対象のデータセットを指定します。この属性は、XRT_TEXT_ATTACH_PIXEL 以外の全ての貼り付け法と一緒に使用できます。

XRT_TEXT_ATTACH_PIXEL_X Position 0
XRT_TEXT_ATTACH_PIXEL_Y Position 0

テキスト・オブジェクトを貼り付ける pixel 座標を指定します。これらの属性は、XRT_TEXT_ATTACH_TYPE が XRT_TEXT_ATTACH_PIXEL の場合にのみ使用されます。

XRT_TEXT_ATTACH_POINT int 0
XRT_TEXT_ATTACH_SET int 0

テキスト・オブジェクトを貼り付けるデータ・ポイント(セットとポイント・インデックス)を指定します。これらの属性は、XRT_TEXT_ATTACH_TYPE が XRT_TEXT_ATTACH_DATA または XRT_TEXT_ATTACH_DATA_VALUE の場合に使用されます。

XRT_TEXT_ATTACH_VALUE_X float 0.0
XRT_TEXT_ATTACH_VALUE_Y float 0.0

テキスト・オブジェクトを貼り付けるグラフの X と Y 値を指定します。これらの属性は、XRT_TEXT_ATTACH_TYPE が XRT_TEXT_ATTACH_VALUE の場合に使用されます。XRT_TEXT_ATTACH_VALUE_Y は、XRT_TEXT_ATTACH_TYPE が XRT_TEXT_ATTACH_DATA_VALUE の場合にも使用されます。

XRT_TEXT_ATTACH_TYPE XrtAttachType(enum) XRT_TEXT_ATTACH_VALUE

テキスト・オブジェクトをどのようにグラフに貼り付けるか指定します。有効な値は、XRT_TEXT_ATTACH_PIXEL、XRT_TEXT_ATTACH_VALUE、XRT_TEXT_ATTACH_DATA、XRT_TEXT_ATTACH_DATA_VALUE です。各貼り付け方法は、更にその他の属性も使用して、実際に貼り付ける位置を決定します。

XRT_TEXT_BACKGROUND_COLOR char* NULL

テキスト・オブジェクトの背景色を指定します。NULL の場合は無色です。ストリングは、色名か#で始まる RGB(赤、緑、青)構成の二桁の 16 進数です。色の指定については、セクション 2.8 をご覧ください。

モノクロのサーバでは、黒以外の色を指定しますと白に表示されます。

XRT_TEXT_BORDER XrtBorder(enum) XRT_TBORDER_NONE

テキスト・オブジェクトの枠線を指定します。有効な値は、XRT_BORDER_NONE、XRT_BORDER_PLAIN、XRT_BORDER_3D_IN、XRT_BORDER_3D_OUT、XRT_BORDER_ETCHED_INXRT_BORDER_ETCHED_OUT、XRT_BORDER_FRAME_IN、XRT_BORDER_FRAME_OUT、XRT_BORDER_BEVEL、XRT_BORDER_SHADOW です。

XRT_TEXT_BORDER_WIDTH Dimension 2

テキスト・オブジェクトの枠線の幅を pixel で指定します。値の範囲は 0 から 20 です。

XRT_TEXT_FONT XmFontList **現在の軸のフォント**

テキスト・オブジェクト内のテキストに使用されるフォントを指定します。

XRT_TEXT_FOREGROUND_COLOR char* NULL

テキスト・オブジェクトの前景色を指定します。NULL の場合は、ウィジェットの前景色が使用されます。ストリングは、色名か#で始まる RGB(赤、緑、青)構成の二桁の 16 進数です。色の指定については、セクション 2.8 をご覧ください。

モノクロのサーバでは、白以外の色を指定しますと黒に表示されます。

XRT_TEXT_IMAGE　　XrtImage　　NULL

テキスト・オブジェクトの背景に表示するイメージのファイル名を指定します。

XRT_TEXT_IMAGE_LAYOUT　　XrtImageLayout　　XRT_IMAGELAYOUT_CENTERED

背景のイメージの位置とサイズを指定します。有効な値は、`XRT_IMAGELAYOUT_CENTERED` か `XRT_IMAGELAYOUT_TILED` です。

XRT_TEXT_IMAGE_MINIMUM_SIZE　　Boolean　　False

イメージのサイズに対応するラベルの最小サイズを指定します。テキストではないイメージ(グラフ領域の背景のイメージ等)には使用されません。

XRT_TEXT_IS_CONNECTED　　Boolean　　False

テキスト・オブジェクトと貼り付けるポイントを結ぶ線を表示するかどうか指定します。この線は、テキスト・オブジェクトの前景色を使用した、1 pixel 幅の線です。

XRT_TEXT_IS_SHOWING　　Boolean　　True

テキスト・オブジェクトの表示、非表示を指定します。

XRT_TEXT_NAME　　char*　　NULL

テキスト・オブジェクトの名前を指定します。これは、アプリケーションに使用され、ウィジェトには使用されません。

XRT_TEXT_OFFSET　　int　　0

テキスト・オブジェクトの中心と貼り付けられるポイントとの間の pixel オフセットを指定します。オフセットの方向は、アンカー(anchor)の方向と同じです。

XRT_TEXT_PRINT_FONT　　char*　　NULL

XRT/graph を *XrtDrawPS()*で印刷する時に、テキスト・オブジェクトに使用されるフォントを指定します。`NULL` の場合は、*XrtDrawPS()*は一番近い PostScript フォントを自動的に選択します。[1] この属性は、どの PostScript フォントにもセットできます。

XRT_TEXT_ROTATION　　XrtRotate(enum)　　XRT_ROTATE_NONE

テキスト・オブジェクトに表示されたテキストの回転を指定します。`XRT_ROTATE_90` または `XRT_ROTATE_270` にセットされた時は、テキストは、時計と反対方向に 90 度または 270 度回転します。

回転されたテキストを使用すると、リアルタイムの速度が落ちますので、秒単位のアップデートが必要なグラフには使用しないでください。

[1] PostScript フォントは、ほとんどの PostScript プリンタで使用できる 35 の PostScript フォントの中から選択できます。(P227 の図 92 参照)

XRT_TEXT_STRINGS	char**	NULL

テキスト・オブジェクトに表示されるストリングの NULL 終端リストを指定します。

XRT_TEXT_WIDTH	Dimension	dynamic
XRT_TEXT_HEIGHT	Dimension	dynamic

テキスト・オブジェクトの高さと幅(pixel)を持ちます。

(注意): この属性は read-only で、テキスト・オブジェクトの高さと幅はセットできません。

XRT_TEXT_X	Position	dynamic
XRT_TEXT_Y	Position	dynamic

テキスト・オブジェクトの左上角の(X、Y)座標を持ちます。

(注意): この属性は read-only です。テキスト・オブジェクトの位置指定には、Attachment(貼り付け)に関する属性を使用してください。

C

PointStyle オブジェクトの属性

PointStyle オブジェクトの全ての属性のアルファベット順の一覧です。属性名の後は、データ型とデフォルト値が記載してあります。

インスタンスの生成: `hPointStyleObject = XrtPointStyleCreate(graph)`

属性の方法: *XrtPointStyleGetValues(), XrtPointStyleSetValues()*

XRT_POINTSTYLE_DATASET int XRT_DATASET1

ポイントにスタイルを適用する場合、どのデータ・セットを使用するか指定します。有効な値は XRT_DATASET1 か XRT_DATASET2 です。

XRT_POINTSTYLE_SET int 0
XRT_POINTSTYLE_POINT int 0

ポイント・スタイルを適用する時に使用するセットとポイントのインデックスを指定します。これらのリソースを使用する前は、XRT_POINTSTYLE_DATASET が必ずセットされているようにしてください。XRT_POINTSTYLE_POINT の有効な値は XRT_POINT_ALL(セット内の全てのポイント、凡例も含む)か、XRT_POINT_LEGEND(凡例の属性のみ)です。

XRT_POINTSTYLE_DISPLAY XrtDisplay (enum) XRT_DISPLAY_SHOW

円グラフのスライスの表示を指定します。有効な値は、XRT_DISPLAY_SHOW、XRT_DISPLAY_HIDE、XRT_DISPLAY_EXCLUDE です。

注意: セットが XRT_DISPLAY_SHOW となっていない場合は、各ポイントを XRT_DISPLAY_SHOW にしても何も変更されません。

XRT_POINTSTYLE_SLICE_OFFSET float 0.0

スライスのオフセットに使用されます。float 値は、円の半径に対するパーセンテージとして指定されます。

XRT_POINTSTYLE_SLICE_OFFSET_USE_DEFAULT　Boolean　True

True の場合、XRT_POINTSTYLE_SLICE_OFFSET は実行時に決定されます。XRT_POINTSTYLE_SLICE_OFFSET に対して明示的に値が与えられている場合は、この Boolean は False にセットされます。

XRT_POINTSTYLE_DATA_STYLE　XrtDataStyle *　dynamic

ポイントのデータ・スタイルに使用されます。

注意: PointStyle に XrtDataStyle をセットすると、指定されたセットとポイントのデータ・スタイルの全ての属性が影響を受けます。

XRT_POINTSTYLE_DATA_STYLE_USE_DEFAULT　Boolean　True

True の場合、XRT_POINTSTYLE_DATA_STYLE は実行時に決定されます。XRT_POINTSTYLE_DATA_STYLE に対して明示的に値が与えられている場合は、この Boolean は False にセットされます。

XRT_POINTSTYLE_PATTERN_BACKGROUND_COLOR　Char *　NULL

ポイントの背景色を指定します。

XRT_POINTSTYLE_USE_DEFAULT　Boolean　True

他の USE_DEFAULT リソースとは異なり、XRT_POINTSTYLE_USE_DEFAULT は、一つのリソースに対応するのではなく、PointStyle オブジェクト・リソースの全てに対応します。

True の場合、PointStyle オブジェクトの全てのリソースの値は XRT/graph によって実行時に決定されます。一方、これらのリソースに明示的に値がセットされている場合は、その Boolean は False にセットされます。

XRT_POINTSTYLE_USE_DEFAULT を明示的に False にセットすると、現在の PointStyle の属性の状態が PointStyle に記憶され、デフォルトとして適用されていた値は、その属性に対する現在値にセットされます。但し、これには二つ例外があり、これらの属性の値は XRT/graph によって動的に決定されます。一つは NULL にセットされている XRT_POINTSTYLE_PATTERN_BACKGROUND_COLOR、もう一つは XRT_DISPLAY_SHOW にセットされている XRT_POINTSTYLE_DISPLAY です。これらの値は、XRT/graph に属性の値を表示する時に計算するよう指示します。

D

手続きと方法

XRT/graphの手続きと方法がアルファベット順に解説されています。“(source)”とある手続きには、ソースコードが提供されてます。ソースのルーチンは、*$XRTHOME/src/graph/util*にあります。ソース・ルーチンを変更する時は、他のディレクトリにコピーしてください。(名前を変更しない場合は、XRT/graphの前にリンクしてください。)

XrtArgValToFloat()

`XtArgVal` を float に変換します。これは、*XrtFloatToArgVal()*と反対の手続きです。

```
float
XrtArgValToFloat(
  XtArgVal value
  )
```

XrtArrCheckAxisBounds()

追加のArrayデータがグラフの再描画を必要とするか調べます。

```
int
XrtArrCheckAxisBounds(
  Widget  graph,
  int     dataset,
  int     numpoints
  )
```

*graph*は、グラフのハンドルです。*dataset*は、データがグラフの一番目または二番目のどちらのデータ・セットに追加されたかを示し、1か2のどちらかです。*numpoints*は、データ・セットの各セットに加えられたポイントの数です。

この手続きを呼び出す前に、グラフ・ウィジェットが既に作成され、`XtNxrtData`か`XtNxrtData2`を使用して接続されたArray型のデータ・オブジェクトを持っている必要があります。

この手続きを使用するには、最初に、配列の各セットに*numpoints*のデータ値を追加します。*XrtArrDataAppendPts()*はこの場合に便利な簡易ルーチンです。ポイントは、必ず、ウィジェットの`XtNxrtData`か`XtNxrtData2`のリソースによって現在指されているデータ・オブジェクトに加えてください。

データを追加したら、*XrtArrCheckAxisBounds()*を呼び出してください。全ての追加データが現在の軸の範囲に収まる場合、このルーチンは0を返します。それ以外の場合は、新しいデータがどの軸を越えるかを示すビット・パターンが返されます。

```
#define XRT_LTX 0x01 /* New data < X axis min */
#define XRT_GTX 0x02 /* New data > X axis max */
#define XRT_LTY 0x04 /* New data < Y axis min */
#define XRT_GTY 0x08 /* New data > Y axis max */
```

戻り値は、scrolling strip chartアプリケーションで、"data window"をいつスクロールするかを決定する時に使用することができます。

XrtArrDataAppendPts() (source)

Arrayデータ・オブジェクトの各セットに新しいポイントを追加します。

```
int
XrtArrDataAppendPts(
  XrtDataHandle    data,
  float        xval,
  float        *yvector
  )
```

*xval*は、新しいX値です。*yvector*は、Y値の配列を指します。*XrtArrDataAppendPts()*は、成功時には1を失敗時には0を返します。前のバージョンとの互換性を保つために、*data*は`XrtData`構造体のポインタとすることもできます。

XrtArrDataFastUpdate()

既存のグラフに、追加のArrayデータを可能な限り速く表示します。

```
int
XrtArrDataFastUpdate(
  Widget  graph,
  int     dataset,
  int     numpoints
  )
```

*graph*は、グラフのハンドルです。*dataset*は、データがグラフの一番目または二番目のどちらのデータ・セットに追加されるかを示し、1か2のどちらかです。*numpoints*は、データ・セットの各セットに加えられたポイントの数です。

この手続きを呼び出す前に、グラフ・ウィジェットが既に作成され、`XtNxrtData`か`XtNxrtData2`を使用して接続されたArray型のデータ・オブジェクトを持っている必要があります。

この手続きを使用するには、最初に、配列の各セットに*numpoints*のデータ値を追加します。*XrtArrDataAppendPts()*はこの場合に便利な簡易ルーチンです。ポイントは、必ず、ウィジェットの`XtNxrtData`か`XtNxrtData2`のリソースによって現在指されているデータ・オブジェクトに加えてください。

追加データを加えたら、*XrtArrDataFastUpdate*()を呼び出してください。新しく加えられたポイントがグラフ化されます。高速アップデートが可能な場合、新しいポイントは瞬時に加えられますが、それ以外の場合は、(`XtNxrtData`か`XtNxrtData2`をリセットしたかのように多少の時間がかかって)グラフが再描画されます。このルーチンは、グラフの再描画が要求された時は0を、高速アップデートが可能な時は1を返します。

*XrtArrCheckAxisBounds()*で、高速アップデートが可能か事前に調べることができます。

XrtArrDataRemovePts() (source)

Array データ・オブジェクトの全てのセットから、*point* でインデックスされているポイントを取り去ります。

```
int
XrtArrDataRemovePts(
  XrtDataHandle    data,
  int          point
  )
```

*XrtArrDataRemovePts()*は、成功時には 1 を、失敗時には 0 を返します。前のバージョンとの互換性を保つために、*data* は XrtData 構造体のポインタとすることもできます。

XrtArrDataShiftPts() (source)

Array データ・オブジェクトのポイントのブロックをコピーします。全てのセットのポイントがシフトされ、Shift-out された(移動によって空白になった)ポイントは、hole 値にセットされます。

```
int
XrtArrDataShiftPts(
  XrtDataHandle data,
  int     dest,
  int     src,
  int     npoints
  )
```

前のバージョンとの互換性を保つために、*data* は XrtData 構造体のポインタとすることもできます。*src* は、シフトされるブロックの最初のポイントを指定します。*npoint* は、ブロックのサイズです。*dest* はブロックのシフト先のポイントです。*src*、*dest* とも 0 から開始されます。

データ・オブジェクトの *npoints* は変更されません。この手続きは、成功時は 1、失敗時は 0 を返します。

XrtCustomFormatValidate()

指定された注釈フォーマットのストリングが有効か調べます。

```
int
XrtCustomFormatValidate(
  char    *format
  )
```

format は、注釈のフォーマットを指定するストリングです。この手続きは、成功時は 0、失敗時は 0 ではない値を返します。

XrtDataConcat() (source)

data1 と *data2* で指されている二つのデータ・セットを連結して作成された、新しいデータ・オブジェクトのハンドルを返します。失敗時には NULL を返します。

```
XrtDataHandle
XrtDataConcat(
  XrtDataHandle    data1,
  XrtDataHandle    data2
)
```

前のバージョンとの互換性を保つため、どちらのパラメータも XrtData 構造体の

ポインタとすることができます。*XrtDataConcat()*は、両方のパラメータが XrtData の場合にのみ、XrtData 構造体を返します。結果の型は、*data1* と *data2* の型に依存します。

data1	data2	結果
NULL	anything	data2
anything	NULL	data1
General	General	General
General	Array	General
Array	General	General
Array	Array	General **または** Array

Array と Array を連結する場合は、両方の Array がセットに対して同じ数のポイントと同じ X 値を持っていれば、Array が返されます。それ以外は、General が返されます。

data1 または *data2* が 0 のセットまたはポイントを持っていると、NULL として処理されます。

XrtDataConvertToHandle()

(前のバージョンで使用されていた)XrtData 構造体を、XrtDataHandle によって参照されるデータ・オブジェクトにコピーします。エラーが起きると NULL ハンドルが戻ります。

```
XrtDataHandle
XrtDataConvertToHandle(
  XrtData *data
  )
```

この手続きは、既存の *data* を開放しません。

XrtDataCopy() (source)

渡された XrtDataHandle オブジェクトの正確なレプリカを返します。

```
XrtDataHandle
XrtDataCopy(
  XrtDataHandle    data
  )
```

*XrtDataCopy()*は、失敗時には NULL を返します。前のバージョンとの互換性を保つために、*data* は XrtData のポインタとすることができ、その場合は、XrtData のポインタが返されます。

XrtDataCreate()

空のデータ・オブジェクトを割り当て、hole 値に XRT_HUGE_VAL を使用します。

```
XrtDataHandle
XrtDataCreate(
  XrtDataType  data_style,
  int      nsets,
  int      npoints
  )
```

data_style は、XRT_DATA_GENERAL または XRT_DATA_ARRAY のどちらかです。スタイルが XRT_DATA_GENERAL の場合、*npoints* は、各セットの最大ポイント数を参照します。

XrtDataCreateFromFile()

データ・オブジェクトを割り当て、テキスト・ファイルから読み込まれたデータをロードします。データ・オブジェクトのハンドルを返します。

```
XrtDataHandle
XrtDataCreateFromFile(
  char*filename,
  char*errbuf
  )
```

*filename*は、データ・ファイルのパス名です。*XrtDataCreateFromFile()*がエラーに遭遇すると NULLを返し、*errbuf*にエラー・メッセージを書きます。*errbuf*がNULLであれば、メッセージは *stderr*に書かれます。*errbuf*は最低限100 byte必要です。

ファイルの"#"で始まる行はコメントとして扱われ、無視されます。最初の、コメントでない行は、"ARRAY"か"GENERAL"で始まり、各セットのセット数(*nsets*)とポイント数(*npoints*)を指定する二つの整数が後に続きます。(GENERALが指定されている場合は、二番目の整数は、各セットの*maximum*ポイント数を指定します。）二つの整数の後に、データがトランスポーズされている事を示す"T"を付けることもできます。

二番めのコメントでない行は、hole値(HOLE <value>)を指定することもできます。XRT/graphは、データ・セットにこの値を見つけると、そのポイントが欠けているとみなします。デフォルトのhole値は、XRT_HUGE_VALです。

ARRAYが指定されている場合は、その次の*npoints*値はX値として扱われます。その後に続く *npoints*は一番目のセットのY値、その後の*npoints*は二番目のセットのY値として扱われます。

ARRAY DATA

ARRAY の指定の例です。

```
# The Michelle's Microchips mm63 file
ARRAY 2 4
HOLE 10000
# X-values
1.0 2.0 3.0 4.0
# Y-values line 1 - Expenses
150.0 175.0 160.0 170.0
# Y-values line 2 - Revenue
125.0 100.0 225.0 300.0
```

ARRAY DATA "T"

同じ ARRAY データでトランスポーズされた例です。

```
# The Michelle's Microchips mm63 file transposed
ARRAY 2 4 T
# X-valuesY1 Expenses  Y2 Revenues
1.0       150.0        125.0
2.0       175.0        100.0
3.0       160.0        225.0
4.0       170.0        300.0
```

GENERAL が指定されている場合、各セットの定義は、セットのポイント数を指定する整数で始まり、次に各セットの X 値、そして各セットの Y 値が来ます。

GENERAL DATA

GENERAL の指定の例です。

```
# 3 lines, max 5 points each
GENERAL 3 5
HOLE 10000
# line 1, 3 points
3
1 3 5
5 14 24
# line 2, 5 points
5
1 2 3 4 5
11 15 18 21 22.
# line 3, 2 points
2
2 4
16 15.2
```

GENERAL DATA"T"

GENERAL データも"T"というトランスポーズのオプションを付けて指定できます。

```
# 3 lines, max 5 points each, transposed
GENERAL 3 5 T
# line 1, 3 points
3
# X   Y
1 5
3 14
5 24
# line 2, 5 points
5
# X   Y
1 11
2 15
3 18
4 21
5 22
# line 3, 2 points
2
# X   Y
2 16
4 15.2
```

XrtDataDestroy()

データ・オブジェクトを破棄します。*XrtDataCreate()*と *XrtDataCreateFromFile()*によって作成されたオブジェクトにのみ使用してください。

```
void
XrtDataDestroy(
  XrtDataHandle    data
  )
```

data は、破棄されるデータ・オブジェクトのハンドルです。グラフに使用されている場合は、自動的に切り離されます。

XrtDataExtractSet() (source)

data で参照されているデータ・オブジェクトの *set* でインデックスされたセットのコピーを持つ、新しいデータ・オブジェクトのハンドルを返します。

```
XrtDataHandle
XrtDataExtractSet(
  XrtDataHandle    data,
  int          set
  )
```

返されたデータ・オブジェクトは、渡されたオブジェクトと同じ型(Array または General)です。*XrtDataExtractSet()*は、失敗時には `NULL` を返します。前のバージョンとの互換性を保つために、*data* は、`XrtData` のポインタとすることができ、その場合は、`XrtData` のポインタが返されます。

XrtDataGetDisplay()

与えられた *set* が表示/非表示か、無視されているか調べます。

```
XrtDisplay
XrtDataGetDisplay(
  XrtDataHandle    data,
  int          set
  )
```

この手続きは、エラーが起きると-1 を返します。セットの表示を変更する場合は、*XrtDataSetDisplay()*を使用してください。

XrtDataGetFirstPoint()

与えられた *set* で、グラフ化される最初のポイントのインデックス(0 から開始)を返します。

```
int
XrtDataGetFirstPoint(
  XrtDataHandle    data,
  int          set
  )
```

set は、Array データ・オブジェクトでは無視されます。エラーが起きると、-1 を返します。

XrtDataGetFirstSet()

与えられたデータ・オブジェクトで、グラフ化される最初のセットのインデックス(0 から開始)を返します。エラーが起きると、-1 を返します。

```
int
XrtDataGetFirstSet(
  XrtDataHandle    data
  )
```

XrtDataGetHole()

与えられたデータ・オブジェクトの hole 値を返します。エラーが起きると、`-XRT_HUGE_VAL` を返します。

```
float
XrtDataGetHole(
  XrtDataHandle    data
  )
```

XrtDataGetLastPoint()

与えられた *set* で、グラフ化される最後のポイントのインデックス(0 から開始)を返します。

```
int
XrtDataGetLastPoint(
  XrtDataHandle    data,
  int          set
  )
```

set は、Array データ・オブジェクトでは無視されます。エラーが起きると、-1 を返します。

XrtDataGetLastSet()

与えられたデータ・オブジェクトで、グラフ化される最後のセットのインデックス(0 から開始)を返します。エラーが起きると、-1 を返します。

```
int
XrtDataGetLastSet(
  XrtDataHandle    data,
  )
```

XrtDataGetNPoints()

与えられた *set* の合計ポイント数を返します。

```
int
XrtDataGetNPoints(
  XrtDataHandle    data,
  int          set
  )
```

set は、Array データ・オブジェクトでは無視されます。エラーが起きると、-1 を返します。

XrtDataGetNSets()

与えられたデータ・オブジェクトの合計セット数を返します。エラーが起きると、-1 を返します。

```
int
XrtDataGetNSets(
  XrtDataHandle    data
  )
```

XrtDataGetType()

与えられたデータ・オブジェクトが Array か General かを調べます。エラーが起きると、-1 を返します。

```
XrtDataType
XrtDataGetType(
  XrtDataHandle    data
  )
```

XrtDataGetXData()

与えられた *set* の X 値の配列の内部ポインタを取得します。これらの X 値は直接変更することができます。この手続きを呼び出すと、次に *XrtDataRelease()*を呼び出すまで、データ・オブジェクトに対する変更を行うことはできません。

```
float *
XrtDataGetXData(
  XrtDataHandle    data,
  int          set
  )
```

set は、Array データ・オブジェクトでは無視されます。エラーが起きると、NULL を返します。

XrtDataGetXElement()

与えられた *set* と *point* の X 値を返します。

```
float
XrtDataGetXElement(
  XrtDataHandle    data,
  int          set,
  int          point
  )
```

set は、Array データ・オブジェクトでは無視されます。エラーが起きると、
-XRT_HUGE_VAL を返します。

XrtDataGetYData()

与えられた *set* の Y 値の配列の内部ポインタを取得します。これらの Y 値は直接変更することができます。この手続きを呼び出すと、次に *XrtDataRelease()*を呼び出すまで、データ・オブジェクトに対する変更を行うことはできません。

```
float *
XrtDataGetYData(
  XrtDataHandle    data,
  int          set
  )
```

エラーが起きると NULL を返します。

XrtDataGetYElement()

与えられた *set* と *point* の Y 値を返します。

```
float
XrtDataGetYElement(
  XrtDataHandle    data,
  int          set,
  int          point
  )
```

この手続きは、エラーが起きると-XRT_HUGE_VAL を返します。

XrtDataRelease()

与えられたデータ・オブジェクトのコントロールを XRT/graph に返します。データ・オブジェクトの方法を呼び出して、属性を設定することができるようになります。*XrtDataGetXData()* または *XrtDataGetYData()*を使用して取得されたポインタは使用できませんので、再度取得し直してください。

```
int
XrtDataRelease(
  XrtDataHandle    data
  )
```

この手続きは、成功時には1、失敗時には0を返します。

XrtDataRemoveSet() (source)

Array または General データ・オブジェクトから、*set* でインデックスされているセットを取り去ります。

```
int
XrtDataRemoveSet(
  XrtDataHandle    data,
  int          set
  )
```

この手続きは、成功時には 1、失敗時には 0 を返します。前のバージョンとの互換性を保つために、*data* は、XrtData のポインタとすることもできます。

XrtDataSaveToFile()

データ・オブジェクトのデータを、*XrtDataCreateFromFile()*で使用できるフォーマットでテキスト・ファイルに書きます。

```
int
XrtDataSaveToFile(
  XrtDataHandle    data,
  char     *filename,
  char     *errbuf
  )
```

data は、書かれるデータ・オブジェクトのハンドルです。*filename* は、書き込みされるファイルのパス名。エラーが起きると、メッセージは *errbuf* に書かれます。(*errbuf* は最低限 100 byte の長さ必要です。) *errbuf* が NULL であれば、メッセージは *stderr* に書かれます。この手続きは、成功時には 1、失敗時には 0 を返します。*XrtDataSaveToFile()*は、データ・オブジェクトの hole 値をファイルの 2 行目に書きます。

XrtDataSetDisplay()

与えられた *set* の表示を変更します。

```
int
XrtDataSetDisplay(
  XrtDataHandle    data,
  int          set,
  XrtDisplay       display
  )
```

display は表示の状態を示し、XRT_DISPLAY_SHOW、XRT_DISPLAY_HIDE、XRT_DISPLAY_EXCLUDE のどれかです。成功時には1、失敗時には0を返します。

HiLo、HiLoOpenClose、キャンドル・グラフの場合は、セットを除外すると棒全体も除外されます。また、high と low のどちらかを非表示にすると棒全体も非表示になり、

open または close のどちらかを非表示にすると、両方とも非表示になります。バブル・チャートでは、セットを除外または非表示にすると、論理的なセット(logical set)全体が除外または非表示になります。

XrtDataSetFirstPoint()

与えられた *set* で、最初にグラフ化されるポイントを指定します。

```
int
XrtDataSetFirstPoint(
  XrtDataHandle    data,
  int          set,
  int          point
  )
```

set は、Array データ・オブジェクトでは無視されます。*point* は、グラフ化する最初のポイントのインデックス。この手続きは、成功時は 1、失敗時は 0 を返します。

XrtDataSetFirstSet()

与えられたデータ・オブジェクトで、最初にグラフ化されるセットを指定します。ここで指定されるセットより前にあるセットは、グラフから除外されます。

```
int
XrtDataSetFirstSet(
  XrtDataHandle    data,
  int          set,
  )
```

set は、グラフ化する最初のセットのインデックスです。この手続きは、成功時は 1、失敗時は 0 を返します。

XrtDataSetHole()

hole(欠けている)と扱われるデータ値を指定します。この手続きは、成功時は 1、失敗時は 0 を返します。

```
int
XrtDataSetHole(
  XrtDataHandle    data,
  float        hole
  )
```

XrtDataSetLastPoint()

与えられた *set* で、最後にグラフ化されるポイントを指定します。

```
int
XrtDataSetLastPoint(
  XrtDataHandle    data,
  int          set,
  int          point
  )
```

set は、Array データ・オブジェクトでは無視されます。*point* は、最後にグラフ化されるポイントのインデックスです。この手続きは、成功時は 1、失敗時は 0 を返します。

XrtDataSetLastSet()

与えられたデータ・オブジェクトで、最後にグラフ化されるセットを指定します。ここで指定されたセットより後にあるセットは、グラフから除外されます。

```
int
XrtDataSetLastSet(
  XrtDataHandle    data,
  int          set,
  )
```

set はグラフ化する最後のセットのインデックスです。この手続きは成功時は 1、失敗時は 0 を返します。

XrtDataSetNPoints()

与えられた *set* のポイント合計数を *npoints* に変更します。(*XrtDataSetFirstPoint()* や *XrtDataSetLastPoint()* を使用して)定義されたデータのサブセットはリセットされます。

```
int
XrtDataSetNPoints(
  XrtDataHandle    data,
  int          set,
  int          npoints
  )
```

set は、Array データ・オブジェクトでは無視されます。*npoints* が以前より大きい場合は、新しいポイントは hole 値にセットされます。成功時は 1、失敗時は 0 を返します。

XrtDataSetNSets()

与えられたデータ・オブジェクトのセット合計数を *nsets* に変更します。(*XrtDataSetFirstSet()* や *XrtDataSetLastSet()* を使用して)定義されたデータのサブセットはリセットされます。

```
int
XrtDataSetNSets(
  XrtDataHandle    data,
  int          nsets
  )
```

nsets が以前より大きい場合、Array データの新しいセットは、hole 値に設定されたポイントでセットされ、General データの新しいセットにはポイントは与えられません。この手続きは、成功時は 1、失敗時は 0 を返します。

XrtDataSetType()

与えられたデータ・オブジェクトのデータ型を変更します。Array データから General データへの変更のみがサポートされています。この手続きは、Array データの共通の X 値を General 型の各セットにコピーします。

```
int
XrtDataSetType(
  XrtDataHandle    data,
  XrtDataType      type
  )
```

type は XRT_DATA_ARRAY か XRT_DATA_GENERAL のどちらかです。この手続きは、成功時は 1、失敗時は 0 を返します。

XrtDataSetXData()

与えられた *set* の X 値をセットします。指定された X 値は、データ・オブジェクトの内部の X 値にコピーされます。

```
int
XrtDataSetXData(
  XrtDataHandle    data,
  int          set,
  float        *xvalues,
  int          n,
  int          index
  )
```

set は、Array データ・オブジェクトでは無視されます。*xvalues* は X 値の配列を指し、これは、使用後解放することができます。*n* は変更するポイントの数で、*index* は変更が開始されるポイントです。この手続きは、成功時は 1、失敗時は 0 を返します。

XrtDataSetXElement()

set と *point* のインデックスにある X 値をセットします。

```
int
XrtDataSetXElement(
  XrtDataHandle    data,
  int          set,
  int          point,
  float        x
  )
```

x は新しい X 値です。*set* は、Array データ・オブジェクトでは無視されます。この手続きは成功時は 1、失敗時は 0 を返します。

XrtDataSetYData()

与えられた *set* の Y 値をセットします。指定された Y 値は、データ・オブジェクトの内部の Y 値にコピーされます。

```
int
XrtDataSetYData(
  XrtDataHandle    data,
  int          set,
  float        *yvalues,
  int          n,
  int          index
  )
```

yvalues は Y 値の配列を指し、これは使用後解放することができます。*n* は変更するポイントの数で、*index* は変更が開始されるポイントです。この手続きは、成功時は 1、失敗時は 0 を返します。

XrtDataSetYElement()

set と *point* のインデックスにある Y 値をセットします。

```
int
XrtDataSetYElement(
  XrtDataHandle    data,
  int          set,
  int          point,
  float        y
  )
```

y は新しい Y 値です。この手続きは、成功時は 1、失敗時は 0 を返します。

XrtDataSort() (source)

増加していく X 値で、各セットのポイントをソートします。

```
int
XrtDataSort(
  XrtDataHandle    data
  )
```

この手続きは、成功時は 1、失敗時は 0 を返します。前のバージョンとの互換性を保つために、*data* は、`XrtData` のポインタとすることもできます。

XrtDataTranspose() (source)

与えられたデータ・オブジェクトのセットとポイントをトランスポーズ(入れ替え)し、新しいデータを持つデータ・オブジェクトのハンドルを返します。

```
XrtDataHandle
XrtDataTranspose(
  XrtDataHandle    data
  )
```

XrtDataUpdateDataValue() (source)

データ・オブジェクトのデータ値を新しいデータ値に変更します。Y 値のみを置き換えます。

```
int
XrtDataUpdateDataValue(
  XrtDataHandle    data,
  float        oldValue,
  float        newValue
  )
```

oldValue はアップデートされるデータ値、*newValue* は新しいデータ値です。この手続きは、成功時は 1、失敗時は 0 を返します。

XrtDeletePointLabel()
XrtDeletePointLabel2()

グラフのポイント・ラベル配列から一つのラベルを削除します。*XrtDeletePointLabel2()*は、複合グラフの二番目のポイント・ラベルのセットに使用されます。

```
Boolean
XrtDeletePointLabel(
  Widget  graph,
  int     index,
  )
```

graph はグラフのハンドルです。*index*(0 から開始)は、削除されるポイント・ラベルのインデックスを指定します。*index* より大きいインデックスのポイント・ラベルは下にシフトされます。これらの手続きは、成功時は`True`、失敗時は`False` を返します。

XrtDeleteSetLabel()
XrtDeleteSetLabel2()

グラフのセット・ラベル配列から一つのラベルを削除します。*XrtDeleteSetLabel2()*は、複合グラフの二番目のセット・ラベルに使用されます。

```
Boolean
XrtDeleteNthSetLabel(
  Widget  graph,
  int     index,
  )
```

graph はグラフのハンドルです。*index*(0 から開始)は、削除されるセット・ラベルのインデックスを指定します。*index* より大きいインデックスのセット・ラベルは、下にシフトされます。これらの手続きは、成功時は`True`、失敗時は`False` を返します。

XrtDestroyData()

`XrtData`構造体を破棄します。*XrtMakeData()*と*XrtMakeDataFromFile()*で作成された構造体だけに使用します。

(注意)　この手続きは、前のバージョンとの互換性を保つためにサポートされています。*XrtDataCreate()*、*XrtDataCreateFromFile()*で作成されたデータ・オブジェクトの破棄には*XrtDataDestroy()*を使用してください。

```
void
XrtDestroyData(
  XrtData *data,
  Boolean all
  )
```

*data*は破棄される構造体です。もし*all*がTRUEであれば、データ値に割り当てられていたスペースも解放されます。*data*がほかのグラフで使用されている時は、自動的に切り離されます。

XrtDrawPS()

グラフのencapsulated PostScript(EPSF-2.0)をファイルに書きます。*XrtOutputPS()*より、なるべくこの手続きを使用してください。*XrtOutputPS()*は、画面のpixelを取得し、PostScriptの*image*オペレータを使って出力しますが、*XrtDrawPS()*は、デバイスから独立したPostScriptオペレータを使用して真のPostScriptファイルを出力します。

*XrtDrawPS()*によって生成されたファイルは、*XrtOutputPS()*で作成されたファイルより小さく、プリンタの解像度を完全に利用して印刷できます。*XrtDrawPS()*は、PostScriptをカスタマイズする幾つかの引数を持っています。

この手続きを呼び出す前に、必ずウィジェットがrealizeされている必要があります。(但し、ディスプレイにマップされている必要はありません。)

```
int
XrtDrawPS(
  Widget  graph,
  FILE*fp,
  char*msg,
  int     inches,
  double  paper_width,
  double  paper_height,
  double  margin,
  int     landscape,
  double  x_offset,
  double  y_offset,
  double  width,
  double  height,
  char*header_font,
  int     header_size,
  char*footer_font,
  int     footer_size,
  char*anno_font,
  int     anno_size,
  char*legend_font,
  int     legend_size,
  int     fill_background,
  int     smart_mono,
  int     showpage
  )
```

*XrtDrawPS()*は、失敗時には0、成功時には1を返します。*graph*は、出力されるグラフのハンドルです。*fp*は、既にオープンされ書き込みができるファイルを指していなければなりません。*XrtDrawPS()*が問題に遭遇した場合は、*msg*(*msg*には最低100 byte割り当てます)にメッセージを返します。*msg*がNULLであれば、エラー・メッセージは*stderr*に書かれます。

*inches*がTRUEであれば、用紙の幅、高さ、マージンに与えられた値とイメージ・オフセットの幅と高さに与えられた値はインチで解釈されます。それ以外の場合は、センチ・メートルが使用されます。

paper_width、*paper_height*と*margin*は必ず指定してください。*margin*は、ページのまわりの余白の幅を定義します。これはページの端まで印刷できないプリンタには必ず使用します。

*landscape*がTRUEであれば、ページが横向きに印刷されます。

*x_offset*と*y_offset*は、用紙の左下角に対応するイメージの左下角のオフセットを指定します。例えば、用紙の下から2インチ、左から3インチの角へイメージの左下角を合わせる時は、*inches*をTRUEにし、*x_offset*を3.0、*y_offset*を2.0に設定します。

*width*と*height*は、イメージのサイズを指定します。両方とも0.0に設定すると、イメージは元の比率を保ちながら最大限に印刷されます。もし二つのパラメータの内一つだけが非0であ

れば、その値が使用されますが元の比率は保たれます。二つとも非0の場合は、その二つの値が適用され、比率が変更されます(イメージは指定されたサイズにあうよう調整されます)。

ヘッダー、フッター、凡例、注釈には、PostScriptフォントが指定できます。NULLが指定された場合、*XrtDrawPS()*は画面で使用されているXフォントについてXサーバに問い合わせ、同様なフォントを選択します。全てのXサーバがフォントについて同じ情報を持っているとは限らないので、異なったXサーバでアプリケーションが実行された場合に異なったPostScriptフォントが使用されることがあります。*XrtDrawPS()*を使用する時は、フォント名をハードコーディングすることができます。(例:*header_font*を"Times-Roman"で指定) また、*XrtDrawPS()*はフォント・サイズのパラメータを無視します[1]。

*fill_background*がFALSEの場合、*XrtDrawPS()*は背景色を白にセットします。モノクロのプリンタで印刷する時は、前景色は黒にセットされます。

*smart_mono*がXRT_PS_COLOR_AS_ISの場合、画面で使用されている色が使用され、XRT_PS_COLOR_MONOの場合は、色は濃淡が異なる灰色の影にマップされます。XRT_PS_COLOR_AUTOの場合は、出力デバイスがカラーかモノクロかを検知し[2]、それに従って、カラーかモノクロをプリントします。この引数は、モノクロのXサーバでは無視されます。

Times-Roman	Helvetica	NewCenturySchlbk-Roman
Times-Bold	Helvetica-Bold	NewCenturySchlbk-Bold
Times-Italic	Helvetica-Oblique	NewCenturySchlbk-Italic
Times-BoldItalic	Helvetica-BoldOblique	NewCenturySchlbk-BoldItalic
Palatino-Roman	Bookman-Light	Helvetica-Narrow
Palatino-Bold	Bookman-Demi	Helvetica-Narrow-Bold
Palatino-Italic	Bookman-LightItalic	Helvetica-Narrow-Oblique
Palatino-BoldItalic	Bookman-DemiItalic	Helvetica-Narrow-BoldOblique
Courier	AvantGarde-Book	Symbol
Courier-Bold	AvantGarde-Demi	ZapfChancery-MediumItalic
Courier-Oblique	AvantGarde-BookOblique	ZapfDingbats
Courier-BoldOblique	AvantGarde-DemiOblique	

図92　35の標準のLaserWriterIIのフォント

*showpage*がTRUEの場合、*XrtDrawPS()*はEPSFファイルの終わりにPostScript *showpage*オペレータを付けます。同じファイル・ポインタを使用して*XrtDrawPS()*へ幾つかの呼び出しを行い、最後の呼び出し以外全て*showpage*をFALSEに設定することにより、1ページに幾つかのグラフを印刷することも可能です。その場合、各グラフを*x_offset*、*y_offset*、*width*、*height*で配置してください。

[1] フォント・サイズのパラメータ(header_size、footer_size、anno_size、legend_size)は、XRT/graph バージョン 2.3 以前の名残で、互換性をサポートするために存在しています。

[2] SPARCprinter 等のモノクロの PostScript プリンタは、間違ってカラー・プリンタと識別してしまいます。これらのプリンタで smart_mono を使用するには、XRT_PS_COLOR_MONO を使用してください。

次のコードは、グラフを標準のレターサイズ用紙に1/2インチのマージンで、(グラフの元の比率を保ちながら)最大限にファイルに出力します。全てのフォントはデフォルトです。

```
char msg[100];
FILE *fp;
fp = fopen("mygraph.ps", "w");
if (fp) {
  XrtDrawPS(mygraph, fp, msg,
    TRUE,          /* Use inches */
    8.5, 11.0, 0.5,     /* 8.5x11, 1/2" margin*/
    FALSE,         /* Portrait */
    0.0, 0.0, 0.0, 0.0, /* Full Size */,
    NULL, 0, NULL, 0,       /* Default Fonts */
    NULL, 0, NULL, 0,
    TRUE,          /* Keep background */
    XRT_PS_COLOR_AS_IS, /* Don't adjust */
    TRUE);     /* Show the page */
}
```

XrtDupDataStyles()

XrtDataStyle 構造体の配列をコピーします。

```
XrtDataStyle **
XrtDupDataStyles(
  XrtDataStyle     **ds
  )
```

*XrtDupDataStyles()*は、コピーされた配列のポインタを返します。

XrtDupStrings()

ストリングの配列をコピーします。

```
char **
XrtDupStrings(
  char**s
  )
```

*XrtDupStrings()*は、コピーされた配列のポインタを返します。

XrtDupValueLabels()

渡された Value ラベルのコピーを返します。

```
XrtValueLabel **
XrtDupValueLabels(
  XrtValueLabel    **label
  )
```

XrtEnableImageType()

XRT/graph に新しいイメージ型を登録します。イメージは、PNG や JPEG の定義済みの型か、ユーザが定義した型を使用することができます。

```
Boolean
XrtEnableImageType(
      XtPointer    class,
      XtPointer    functions
      )
```

class はウィジェット・クラス(xtXrtGraphWidgetClass)、*functions* はXrtPDSImageFuncs構造体のポインタです。(詳細は、P144 を参照してください。)

XrtFloatToArgVal()

float 値を XtArgVal に変換します。*XtSetValues()*または *XtVaSetValues()*によって、XRT/graphの浮動小数点リソースをセットする時に使用します。*XrtSetValues()*と一緒に使用することはできません。

```
XtArgVal
XrtFloatToArgVal(
  float   value
  )
```

XrtFreeDataStyles()

XrtDataStyle 構造体の配列に使用されていたメモリを解放します。

```
void
XrtFreeDataStyles(
  XrtDataStyle     **ds
  )
```

XrtFreePropString()

*XrtGetPropString()*によって割り当てられたストリングが使用しているメモリを解放します。

```
void
XrtFreePropString(
  char*str
  )
```

XrtFreeStrings()

ストリング配列に使用されていたメモリを解放します。

```
void
XrtFreeStrings(
  char**s
  )
```

XrtFreeTextHandles()

テキスト領域のハンドルの配列に使用されていたメモリを解放します。

```
void
XrtFreeTextHandles(
  XrtTextHandle    *list
  )
```

XrtFreeValueLabels()

Value ラベルのセットに使用されていたメモリを解放します。

```
void
XrtFreeValueLabels(
  XrtValueLabel    **label
  )
```

XrtGenCheckAxisBounds()

追加されたGeneralデータがグラフの再描画を必要とするか調べます。

```
int
XrtGenCheckAxisBounds(
  Widget  graph,
  int     dataset,
  int     set,
  int     numpoints
  )
```

*graph*は、グラフのハンドルです。*dataset*は、データがグラフの一番目または二番目のどちらのデータセットに追加されるかを示し、1か2のどちらかです。*numpoints*は、*set*でインデックスされているセットに加えられたポイントの数です。

この手続きを呼び出す前に、グラフ・ウィジェットが既に作成され、XtNxrtDataかXtNxrtData2のリソースを使用して接続されたGeneral型のデータ・オブジェクトを持っている必要があります。この手続きを使用するには、最初にGeneralデータ・セットの一つのセットに*numpoints*のデータ値を追加ます。*XrtGenDataAppendPt()*は、この場合に便利な簡易ルーチンです。ポイントは、必ず、ウィジェットのXtNxrtDataかXtNxrtData2のリソースによって現在指されているデータ・オブジェクトに加えてください。

データを追加したら、*XrtGenCheckAxisBounds()*を呼び出してください。全ての追加データが現在の軸の範囲に収まる場合、このルーチンは0を返します。それ以外の場合は、新しいデータがどの軸を越えるかを示すビットパターンが返されます。

```
#define XRT_LTX 0x01 /* New data < X axis min */
#define XRT_GTX 0x02 /* New data > X axis max */
#define XRT_LTY 0x04 /* New data < Y axis min */
#define XRT_GTY 0x08 /* New data > Y axis max */
```

戻り値は、scrolling strip chartアプリケーションで"data window"をいつスクロールするかを決定する時に使用できます。

XrtGenDataAppendPt() (source)

set でインデックスされているセットの最後に、(x, y)値の *xval* と *yval* を追加します。

```
int
XrtGenDataAppendPt(
  XrtDataHandle    data,
  int          set,
  float        xval,
  float        yval
  )
```

前のバージョンとの互換性を保つために、*data*はXrtData構造体のポインタとすることもできます。*XrtGenDataAppendPt()*は、成功時には1、失敗時には0を返します。*data*は必ずGeneral型のデータ構造体を指すようにしてください。

XrtGenDataFastUpdate()

既存のグラフに、追加のGeneralデータを可能な限り速く表示します。

```
int
XrtGenDataFastUpdate(
  Widget  graph,
  int     dataset,
  int     set,
  int     numpoints
  )
```

*graph*は、グラフのハンドルです。*dataset*は、データがグラフの一番目または二番目のどちらのデータ・セットに追加されるかを示し、1か2のどちらかです。*numpoints*は、*set*でインデックスされているセットに加えられたポイントの数です。

この手続きを呼び出す前に、グラフ・ウィジェットが既に作成され、XtNxrtDataかXtNxrtData2のリソースを使用して接続されたGeneral型のデータ・オブジェクトを持っている必要があります。

この手続きを使用するには、最初に、Generalデータ・セットの一つのセットに*numpoints*のデータ値を追加します。*XrtGenDataAppendPt()*は、この場合に便利な簡易ルーチンです。ポイントは、必ず、ウィジェットのXtNxrtDataかXtNxrtData2のリソースによって現在指されているデータ・オブジェクトに加えてください。追加データを加えたら、*XrtGenDataFastUpdate*()を呼び出してください。新しく加えられたポイントがグラフ化されます。高速アップデートが可能な場合、新しいポイントは瞬時に加えられますが、それ以外の場合は、(XtNxrtDataかXtNxrtData2をリセットしたかのように多少の時間がかかって)グラフが再描画されます。このルーチンは、グラフの再描画が要求された時は0を、高速アップデートが可能な時は1を返します。

*XrtGenCheckAxisBounds()*で、高速アップデートが可能か事前に調べることができます。

XrtGenDataRemovePt() (source)

Generalデータ・オブジェクトから、*set*と*point*でインデックスされている値を取り去ります。

```
int
XrtGenDataRemovePt(
  XrtDataHandle    data,
  int          set,
  int          point
  )
```

*XrtGenDataRemovePt()*は、成功時には１、失敗時には0を返します。前のバージョンとの互換性を保つために、*data*はXrtData構造体のポインタとすることもできます。

XrtGenDataShiftPts() (source)

Generalデータ・オブジェクトのセットのポイントのブロックをコピーします。Shift-outされた(移動によって空白になった)ポイントは、hole値にセットされます。

```
int
XrtGenDataShiftPts(
  XrtDataHandle    data,
  int          set,
  int          dest,
  int          src,
  int          npoints
  )
```

前のバージョンとの互換性を保つために、*data* は XrtData 構造体のポインタとすることもできます。*set* は、ポイントがシフトされるセットです。*src* は、シフトされるブロックの最初のポイントを指定します。*npoint* は、ブロックのサイズです。*dest* はブロックのシフト先のポイントです。*set*、*src*、*dest* は 0 から開始されます。

データ・オブジェクトのセットの *npoints* は変更されません。この手続きは、成功時は 1、失敗時は 0 を返します。

XrtGetImage()

グラフのXImageのポインタを返します。これは、グラフのXImageにアクセスする必要のあるXlibレベルのプログラミングに便利です。

```
XImage *
XrtGetImage(
  Widget  graph
  )
```

*graph*は出力されるグラフのハンドルです。*XrtGetImage()*は、失敗時にNULLを返します。グラフが画面上で部分的にしか表示されていない場合でもXImageは完成します。

XrtGetNthDataStyle()
XrtGetNthDataStyle2()

グラフに現在使用されている*n*番目のデータ・スタイルのポインタを返します。

```
XrtDataStyle *
XrtGetNthDataStyle(
  Widget  graph,
  int     index
  )
```

*graph*は、グラフのハンドルです。*index*(0から開始)は、どのデータ・スタイルを返すか指定します。*index*が、現在定義されているデータ・スタイルの数より大きい場合は、*index*のデフォルトのデータ・スタイルが返されます。この方法で返されたポインタは、read-onlyです。*XrtGetNthDataStyle2()*は、複合グラフの二番目のセットに使用されます。

XrtGetNthFooterString()

グラフに現在使用されている、*n* 番目のフッター・ストリングのポインタを返します。

```
char *
XrtGetNthFooterString(
  Widget  graph,
  int     index
  )
```

graph は、グラフのハンドルです。*index*(0 から開始)は、どのフッター・ストリングを返すか指定します。*index* が、現在定義されているフッター・ストリングの数より大きい場合は、NULL が返されます。この方法で返されたポインタは read-only です。

XrtGetNthHeaderString()

グラフに現在使用されている、*n* 番目のヘッダー・ストリングのポインタを返します。

```
char *
XrtGetNthHeaderString(
  Widget  graph,
  int     index
  )
```

graph は、グラフのハンドルです。*index*(0 から開始)は、どのヘッダー・ストリングを返すか指定します。*index* が、現在定義されているヘッダー・ストリングの数より大きい場合は、NULL が返されます。この方法で返されたポインタは read-only です。

XrtGetNthPointLabel()
XrtGetNthPointLabel2()

現在グラフで使用されている*n*番目のポイント・ラベルのポインタを返します。

```
char *
XrtGetNthPointLabel(
  Widget  graph,
  int     index
  )
```

*graph*は、グラフのハンドルです。*index*(0から開始)は、どのポイント・ラベルを返すか指定します。*index*が、現在定義されているポイント・ラベルの数より大きい場合は、NULLが返されます。この方法で返されたポインタは、read-onlyです。

*XrtGetNthPointLabel2()*は、Y2がprimary(一番目の)軸となっているグラフのポイント・ラベルの二番目のセットに使用されます。

XrtGetNthSetLabel()
XrtGetNthSetLabel2()

現在グラフで使用されている*n*番目のセット・ラベルのポインタを返します。

```
char *
XrtGetNthSetLabel(
  Widget  graph,
  int     index
  )
```

*graph*は、グラフのハンドルです。*index*(0から開始)は、どのセット・ラベルを返すか指定します。*index*が、現在定義されているセット・ラベルの数より大きい場合は、NULLが返されます。この方法で返されたポインタは、read-onlyです。

*XrtGetNthSetLabel2()*は、複合グラフのセット・ラベルの二番目のセットに使用されます。

XrtGetOtherSliceSets()

与えられたポイントに対して、どのセットが、円グラフの"その他"のスライスにグループ化されるかを調べます。(-1 で終了する)正の整数のリストのポインタが返されます。

```
int *
XrtGetOtherSliceSets(
  Widget  graph,
  int     point,
  int     dataset
  )
```

graph は、グラフのハンドルです。*point*(0 から開始)は、どのセットがグループ化されているかを調べる時に使用するポイントを指定します。*dataset* は、XRT_DATASET1 か XRT_DATASET2 のどちらかです。リストは使用後、解放してください。

XrtGetPropString()

(逆変換プログラムを使用して)グラフのリソースの現在値をストリングとして取得します。XtConvertAndStore をバイパスする時に便利です。

```
Boolean
XrtGetPropString(
  Widget       graph,
  XrtProperty      resource,
  char     **str /* Returned */
  )
```

graph はグラフのハンドルです。*resource* は、(P267 のリストにある、クロス・プラットフォーム名を使用した)リソースを指定します。*str* は、リソース値を表す、返されたストリングのポインタです。*str* は、使用後、*XrtFreePropString()*を使用して必ず解放してください。

XrtGetTextHandles()

グラフに現在定義されているテキストのハンドルを調べます。*XrtGetTextHandles()*は、貼り付けられているテキスト領域の合計数を返します。

```
int
XrtGetTextHandles(
  Widget       graph,
  XrtTextHandle    **list /* Returned */
  )
```

graph は、グラフのハンドルです。*list* は、返されたテキストのハンドルの NULL 終端リストです。このリストは、使用後、*XrtFreeTextHandles()*を使用して必ず解放してください。

XrtGetValueLabel()

軸に指定したラベルに一番近い Value ラベルを調べます。

```
XrtValueLabel *
XrtGetValueLabel(
  Widget       graph,
  XrtAxis      axis,
  XrtValueLabel    *label
  )
```

graph は、グラフのハンドルです。*axis* は、XRT_AXIS_X、XRT_AXIS_Y、XRT_AXIS_Y2 のどれかです。*label* で指されている XrtValueLabel 構造体の *value* は、この手続きが呼ばれる前にセットされている必要があります。

この手続きは、既存の XrtValueLabel 構造体のポインタ、または(パラメータが無効な場合は)NULL を返します。この構造体の *value* と *string* が、(渡されたラベルに)一番近い Value ラベルです。軸に Value ラベルが存在しない場合は、*string* は NULL になります。ここで返されたポインタは、read-only です。

XrtGetValues()

グラフのリソースの現在値を取得します。

```
void
XrtGetValues(
  Widget  graph,
  ...,
  NULL
  )
```

graph は、グラフのハンドルです。...は、(一つまたは複数の)*resource* と *value* のペアです。*resource* は、(P267 のリストにある、XrtProperty クロス・プラットフォーム名で指定される)リソースです。*value* は、返されたリソース値を持つ変数のポインタです。*value* のデータ型は、リソースのデータ型と同じにしてください。float 型のリソースにアクセスする時は、*XrtArgValToFloat()*を使用しないでください。*resource* と *value* のペアのリストは、NULL で終了します。

XrtInsertNthDataStyle()

n 番目のデータ・スタイルになる、新しいデータ・スタイルを挿入します。

```
int
XrtInsertNthDataStyle(
  Widget       graph,
  int          index,
  XrtDataStyle     *ds
  int          dataset
)
```

graph は、グラフのハンドルです。*index*(0 から開始)は、新しいデータ・スタイルのインデックスを指定します。*index* より大きいインデックスのデータ・スタイルは全て上にシフトされます。*ds* は、新しいデータ・スタイルを持つ XrtDataStyle 構造体を指します。XRT/graph は、*ds* によって指されるデータ・スタイルの独自のコピーを作成します。*dataset* は、一番目か二番目のどちらのデータ・スタイルのセットに使用されるかを指定し、1 か 2 です。この方法による副作用で、該当する XtNxrtDataStylesUseDefault または XtNxrtDataStyles2UseDefault が False にセットされます。

XrtInsertNthPointLabel()
XrtInsertNthPointLabel2()

n 番めのポイント・ラベルになる、新しいポイント・ラベルを挿入します。*XrtInsertNthPointLabel2()*は、複合グラフで二番目のセットに使用されます。

```
Boolean
XrtInsertNthPointLabel(
  Widget  graph,
  int     index,
  char*s
  )
```

graph は、グラフのハンドルです。*index*(0 から開始)は、新しいポイント・ラベルのインデックスを指定します。*index* より大きいインデックスのポイント・ラベルは、全て上にシフトされます。*s* は、使用されるストリングです。

XRT/graph は、*s* によって指されている文字の独自のコピーを作成します。これらの手続きは、成功時は`True`、失敗時は`False` を返します。

XrtInsertNthSetLabel()
XrtInsertNthSetLabel2()

n 番めのセット・ラベルになる、新しいセット・ラベルを挿入します。*XrtInsertNthSetLabel2()*は、複合グラフで二番目のセットに使用されます。

```
Boolean
XrtInsertNthSetLabel(
  Widget  graph,
  int     index,
  char*s
  )
```

graph は、グラフのハンドルです。*index*(0 から開始)は、新しいセット・ラベルのインデックスを指定します。*index* より大きいインデックスのセット・ラベルは、全て上にシフトされます。*s* は、使用されるストリングです。XRT/graph は、*s* によって指されている文字の独自のコピーを作成します。これらの手続きは、成功時は`True`、失敗時は`False` を返します。

XrtMakeData()

空の`XrtData`を割り当てる手続きです。`XRT_HUGE_VAL`をhole値として使用します。

(注意): この手続きは、前のバージョンとの互換性を保つためにサポートされています。データ・オブジェクトの割り当てには、*XrtDataCreate()*を使用してください。

```
XrtData *
XrtMakeData(
  XrtDataType      data_type,
  int          nsets,
  int          npoints,
  Boolean      all
  )
```

*data_style*は、`XRT_DATA_GENERAL`か`XRT_DATA_ARRAY`です。*all*が`TRUE`であれば、データ値のスペースも割り当てられますが、それ以外の場合は構造体とポインタだけが割り当てられます。スタイルが`XRT_DATA_GENERAL`の時、*npoints*は各セットの最大ポイント数を参照します。

XrtMakeDataFromFile()

`XrtData`を割り当て、テキスト・ファイルから読み込まれたデータをロードする手続きです。

(注意): この手続きは、前のバージョンとの互換性を保つためにサポートされています。データ・ファイルの読み込みには、*XrtDataCreateFromFile()*を使用してください。

```
XrtData *
XrtMakeDataFromFile(
  char*filename,
  char*errbuf
  )
```

*filename*は、データ・ファイルのパス名です。*XrtMakeDataFromFile()*がエラーに遭遇すると、`NULL`を返し、*errbuf*にエラー・メッセージを書きます。*errbuf*が`NULL`であれば、メッセージは*stderr*に書かれます。*errbuf*は最低限100 byte必要です。

この手続きが理解できるフォーマットに関しては、*XrtDataCreateFromFile()*を参照してください。

XrtMakeTime()

時間の情報を long の整数(1970 年 1 月 1 日からの秒数)に変換します。

```
long
XrtMakeTime(
  int     yr,
  int     mon,
  int     day,
  int     hr,
  int     min,
  int     sec,
  )
```

yr は 1900 年を引いた(二桁の)年です。*mon* は月を表す 0 から 11 の値で、*day* はその月の日です。*hr* は一日の時間を表す、0 から 23 の値です。*min* と *sec* は分と秒です。

XrtMap()

pixel座標をグラフ座標にマップします。

```
XrtRegion
XrtMap(
  Widget       graph,
  int          yaxis,
  int          pix_x,
  int          pix_y,
  XrtMapResult     *map
  )
```

アプリケーションで、*pix_x*と*pix_y*に対応する、(*graph*で指定されるグラフの)
グラフ座標を決定します。一番目のY軸で定義される座標系には*yaxis*を1に設定し、二番目のY軸の座標系なら2に設定します。結果は*map*に返されます。この手続きは、通常、イベント処理手続きに使用されます。

XrtMrmInitialize()

Mrm に XRT/graph ウィジェットのクラスを登録します。UIL アプリケーションに使用されます。このルーチンは、*MrmInitialize()*の後に呼び出してください。

```
void
XrtMrmInitialize(void)
```

XrtOutputPS()

ファイルにグラフのencapsulated PostScript(EPSF)を出力します。PostScriptファイルは、PostScriptプリンタに送ったり、他のPostScriptファイルにインポートする場合に適しています。ファイルは、EPSF-2.0フォーマットに準拠しています。

```
int
XrtOutputPS(
  Widget  graph,
  FILE*fp,
  Boolean color,
  char*msg
  )
```

*graph*は、出力されるグラフのハンドルです。*fp*は、既にオープンされ書き込みできるファイルを指していなければなりません。色がTRUEなら、PostScriptの*colorimage*が作成されますが、それ以外の場合はモノクロのPostScript *image*が作成されます。*XrtOutputPS()*が障害に遭遇すると、*msg*(*msg*には最低100 byte割り当てます)にメッセージを返します。*msg*がNULLであれば、エラー・メッセージは*stderr*に書かれます。*XrtOutputPS()*は、失敗時に0を返し、成功時には1を返します。グラフが画面上で部分的に表示されていない場合でも、PostScriptファイルは完成します。カラーのグラフのPostScriptファイルはサイズが大きいので、このルーチンを呼び出す前に、充分なスペースがあるか確認してください。

XrtOutputRaster()

グラフのSun rasterfileをファイルに出力します。この手続きは、Sun *rasterfile(5)*ルーチンを使用しますので、SPARCアーキテクチャのみでサポートされています。rasterfileは標準のフォーマットで、全てのカラーマップの情報が含まれています。

```
int
XrtOutputRaster(
  Widget  graph,
  FILE*fp,
  char*msg
  )
```

*graph*は、出力されるグラフのハンドルです。*fp*は、既にオープンされ、書き込み可能なファイルを指していなければなりません。*XrtOutputRaster()*が問題に遭遇すると、メッセージを*msg*(*msg*には最低100 byte割り当てます)に返します。*msg*がNULLであると、エラー・メッセージは*stderr*に書かれます。*XrtOutputRaster()*は、失敗時に0、成功時には1を返します。グラフが画面上で部分的に表示されていない場合でも、rasterfileは完成します。カラーのグラフのファイルはサイズが大きいので、このルーチンを呼び出す前に、充分なスペースがあるか確認してください。

XrtOutputXwd()

グラフのX Window Dumpをファイルに出力します。

```
int
XrtOutputXwd(
  Widget  graph,
  FILE*fp,
  char*msg
  )
```

*graph*は、出力されるグラフのハンドルです。*fp*は、既にオープンされ書き込みできるファイルを指していなければなりません。*XrtOutputXwd()*が問題に遭遇すると、

メッセージを*msg*(*msg*には最低100 byte割り当てます)に返します。*msg*がNULLであれば、エラー・メッセージは*stderr*に書かれます。*XrtOutputXwd()*は失敗時に0を返し、成功時には1を返します。グラフが画面上で部分的に表示されていない場合でも、xwdファイルは完成します。

XrtPick()

与えられたpixel座標に一番近く表示されているデータをピックします。

```
XrtRegion
XrtPick(
  Widget       graph,
  XrtDsGroup       ds_group,
  int          pix_x,
  int          pix_y,
  XrtPickResult    *pick,
  XrtFocus     focus
  )
```

アプリケーションで、(*graph*で指定されるグラフの)どのデータ(*set*、*point*)がpixel(*pix_x*、*pix_y*)に一番近く表示されているか調べます。その結果は*pick*に返されます。この手続きは、通常イベント処理手続きに使用されます。

*ds_group*パラメータは、複合グラフでどちらのデータセットを使用するか*XrtPick()*に伝えます。(複合でない)シングル・グラフで*XrtPick()*を呼び出す時は、これをXRT_DATASET1に設定してください。複合グラフでは、XRT_DATASET1かXRT_DATASET2に設定してピックの結果がどちらかのデータ・セットに限定されるようにしてください。XRT_DATASET1 | XRT_DATASET2に設定すると、*XrtPick()*は両方のデータ・セットを対象とします。

*focus*パラメータは、*XrtPick()*ルーチンが一番近いセットとポイントを決定する方法を示します。XRT_XFOCUSにセットすると、X軸に沿った距離が"近さ"の決定に使用されます。XRT_YFOCUSにセットすると、Y軸に沿った距離が使用されます。
XRT_XFOCUS | XRT_YFOCUSに設定するとXとYの両方の軸に沿った距離が使用されます。

XrtPointStyleCreate()

PointStyle オブジェクトをグラフの子として作成します。PointStyle は作成されても、特定なデータ・ポイントには割り当てられません。この方法は、その他の XrtPointStyle...に使用される XrtPointStyleHandle を返します。

```
XrtPointStyleHandle
XrtPointStyleCreate(
      Widget       graph
      )
```

graph はグラフ・ハンドルです。*XrtPoinstStyleCreate()*を除く、全ての PointStyle API ファンクションの一番目のパラメータは、この *XrtPointStyleCreate()* によって返される XrtPoinstStyleHandle です。

XrtPointStyleDestroy()

PointStyle を破棄します。

```
void
XrtPointStyleDestroy(
      XrtPointStyleHandle   pointstyle
      )
```

pointstyle は、PointStyle オブジェクトのハンドルです。

XrtPointStyleGetValues()

PointStyle オブジェクトの属性の現在値を取得します。

```
void
XrtPointStyleGetValues(
      XrtPointStyleHandle   pointstyle,
      ...
      NULL
      )
```

pointstyle は PointStyle オブジェクト・ハンドル。..は、(一つまたは複数の)*attribute* と *value* のペアです。*attritube* は PointStyle オブジェクトの属性、*value* は返された属性値を持つ変数のポインタです。*value* のデータ型は、属性のデータ型と同じです。*attribute* と *value* のペアのリストは必ず NULL で終了するようにしてください。

XrtPointStyleHandle()

セットとポイントの属性に基づいて PointStyle を検索し、これらの属性で最も最近に作成された PointStyle を返します。適合する PointStyle が見つからなければ、NULL を返します。

```
XrtPointStyleHandle
XrtPointStyleFindExact(
      Widget   graph,
      int dataset,
      int set,
      int point
      )
```

graph はグラフ・ハンドル、*dataset* は XRT_DATASET1 か XRT_DATASET2 です。*set* は検索される PointStyle のセットの属性、*point* はポイントの属性です。

XrtPointStyleSetValues()

PointStyle オブジェクトの属性の現在値をセットします。

```
void
XrtPointStyleSetValues(
      XrtPointStyleHandle   pointstyle,
      ...
      NULL
      )
```

pointstyle は PointStyle オブジェクト・ハンドル。..は、(一つまたは複数の)*attribute* と *value* のペアです。*attritube* は PointStyle オブジェクトの属性、*value* は新しい値です。*value* のデータ型は、属性のデータ型と同じです。*attribute* と *value* のペアのリストは必ず NULL で終了するようにしてください。

XrtPropertyEditorEnable()

XRT/graphのProperty Editorをイニシャライズし、特定なグラフに使用できる機能を指定します。

```
Boolean
XrtPropertyEditorEnable(
  Widget  graph,
  Boolean enable,
  char*title_string,
  Boolean allow_load_save,
  Boolean do_updates
  )
```

*graph*は、グラフのハンドルです。*enable*は、Property Editorを使用可(enable)にするか使用不可(disable)にするか指定します。*title_string*は、タイトル・バーに表示されるテキストで、デフォルトでは、”XRT/graph Property Editor”と表示されます。*allow_load_save*は、エンドユーザが、.XRTファイルとデータ・ファイルをロード/保管し、Cコードを生成することができるか指定します。*do_update*は、グラフがアップデートされた時に、Property Editorもアップデートするかどうか指定します。この手続きは、成功時は`True`、失敗時は`False`を返します。

XrtRemoveNthDataStyle()

*n*番目のデータ・スタイルを削除します。

```
int
XrtRemoveNthDataStyle(
  Widget  graph,
  int     index,
  int     dataset
  )
```

*graph*は、グラフのハンドルです。*index*(0から開始)は、削除されるデータ・スタイルのインデックスを指定します。*index*より大きいインデックスのデータ・スタイルは全て下にシフトされます。*dataset*は、一番目か二番目のどちらのセットに使用されるかを指定し、1か2です。この手続きは、成功時は1、失敗時は0を返します。

XrtSamleColorFunc()

`XtNxrtColorFunc`リソースに使用できるファンクションの例です。XPMで一番近い色をマッチさせるアルゴリズムの応用です。このファンクションのソースコードは、*$XRTHOME/src/graph/util/x_color_func.c*にあります。

```
unsigned long
XrtSampleColorFunction(
      Display     *display,
      int     screen,
      Colormap    colormap,
      Xcolor      *color,
      Boolean     do_free
      )
```

XrtSaveDataToFile()

`XrtData`構造体にあるデータを、*XrtMakeDataFromFile()*に適するフォーマットでテキスト・ファイルに書きます。

(注意) この手続きは、前のバージョンとの互換性を保つためにサポートされています。データ・ファイルの保管には、*XrtDataSaveToFile()*を使用してください。

```
int
XrtSaveDataToFile(
  XrtData      *data,
  char     *filename,
  char     *errbuf
  )
```

*data*は、書かれるデータを持った構造体です。*filename*は、書かれるファイルのパス名です。エラーが起きると、メッセージは*errbuf*(最低限100 byte必要です)に書かれます。*errbuf*がNULLであれば、メッセージは*stderr*に書かれます。この手続きは成功時には1、失敗時には0を返します。*XrtSaveDataToFile()*は、XrtData構造体にあるhole値をファイルの二行目に書きます。

XrtSaveImageAsJpeg()

グラフをJPEGに出力します。この方法を使用する前には、*XrtEnableImageType()*を使用して、JPEGのイメージ型をXRT/graphに登録してください。

```
Boolean
XrtSaveImageAsJpeg(
      Widget   graph,
      String   filename,
      String   msg
      )
```

graph は出力されるグラフのハンドル、*filename* はイメージが書かれるファイルの名前です。*XrtSaveImageAsJpeg()*が問題に遭遇した場合は、*msg*(最低 100byte)にメッセージを返します。*msg* が NULL の場合、エラー・メッセージは *stderr* に書かれます。*XrtSaveImageAsJpeg()*は成功時にはTrue を返します。

XrtSaveImageAsPNG()

グラフをPNGに出力します。この方法を使用する前には、*XrtEnableImageType()*を使用して、PNGのイメージ型をXRT/graphに登録してください。

```
Boolean
XrtSaveImageAsPng(
      Widget   graph,
      String   filename,
      String   msg
      )
```

graph は出力されるグラフのハンドル、*filename* はイメージが書かれるファイルの名前です。*XrtSaveImageAsJpeg()*が問題に遭遇した場合は、*msg*(最低 100byte)にメッセージを返します。*msg* が NULL の場合、エラー・メッセージは *stderr* に書かれます。*XrtSaveImageAsPng()*は成功時にはTrue を返します。

XrtSaveImageGeneric()

グラフを指定されたイメージ型のファイルに出力します。(イメージ型が XPM や XWD 等既に登録されている型でない場合は)この方法を使用する前に、*XrtEnableImageType()*によって使用するイメージ型を XRT/graph に登録してください。

```
Boolean
XrtSaveImageGeneric(
      Widget   graph,
      int  image_type
      String   filename,
      String   msg
      )
```

graph は出力されるグラフのハンドル、*image_type* はイメージ型、*filename* はイメージが書かれるファイルの名前です。*XrtSaveImageGeneric()*が問題に遭遇した場合は、*msg*(最低 100byte)にメッセージを返します。*msg* が NULL の場合、エラー・メッセージは *stderr* に書かれます。*XrtSaveImageGeneric()*は成功時には True を返します。

XrtSetNthDataStyle()
XrtSetNthDataStyle2()

*n*番目のデータ・スタイルをセットします。*XrtSetNthDataStyle2()*は、複合グラフで二番目のセットに使用されます。

```
void
XrtSetNthDataStyle(
  Widget       graph,
  int          index,
  XrtDataStyle     *ds
  )
```

graph はグラフのハンドルです。*index*(0 から開始)は、どのデータ・スタイルを変更するか指定します。*ds* は、新たに使用されるデータ・スタイルを持つ XrtDataStyle 構造体を指します。XRT/graph は、*ds* によって指されているデータ・スタイルのコピーを作成します。

この方法を使用すると、副作用で、該当するXtNxrtDataStylesUseDefault または XtNxrtDataStyles2UseDefaultがFalseにセットされます。

XrtSetNthFooterString()

n 番目のフッター・ストリングをセットします。

```
void
XrtSetNthFooterString(
  Widget  graph,
  int     index,
  char*s
  )
```

graph はグラフのハンドルです。*index*(0 から開始)は、どのフッター・ストリングを変更するか指定します。*s* は、新しいストリングです。XRT/graph は、*s* で指されている文字のコピーを作成します。

XrtSetNthHeaderString()

n 番目のヘッダー・ストリングをセットします。

```
void
XrtSetNthHeaderString(
  Widget  graph,
  int     index,
  char*s
  )
```

graph はグラフのハンドルです。*index*(0 から開始)は、どのヘッダー・ストリングを変更するか指定します。*s* は、新しいストリングです。XRT/graph は、*s* で指されている文字のコピーを作成します。

XrtSetNthPointLabel()
XrtSetNthPointLabel2()

n 番目のポイント・ラベルをセットします。ポイント・ラベルは、XtNxrtXAnnotationMethodがXRT_ANNO_POINT_LABELSの場合にのみ表示されます。*XrtSetNthPointLabel2()*は、複合グラフで二番目のセットに使用されます。

```
void
XrtSetNthPointLabel(
  Widget  graph,
  int     index,
  char*s
  )
```

*graph*は、グラフのハンドルです。*index*(0から開始)は、どのポイント・ラベルを変更するか指定します。*s*は、新たに使用されるストリングです。XRT/graphは、*s*によって指されている文字のコピーを作成します。

XrtSetNthSetLabel()
XrtSetNthSetLabel2()

*n*番目のセット・ラベルをセットします。*XrtSetNthSetLabel2()*は、複合グラフで二番目のセットに使用されます。

```
void
XrtSetNthSetLabel(
  Widget  graph,
  int     index,
  char*s
  )
```

*graph*は、グラフのハンドルです。*index*(0から開始)は、どのセット・ラベルを変更するか指定します。*s*は、新たに使用されるストリングです。XRT/graphは、*s*で指されている文字のコピーを作成します。

XrtSetPropString()

グラフのリソースをストリングで表される値にセットします。

```
Boolean
XrtSetPropString(
  Widget       graph,
  XrtProperty      resource,
  char     *str
  )
```

*graph*は、グラフのハンドルです。*resource*は、(P267のリストにある、クロス・プラットフォーム名を使用した)XRT/graphのリソースを指定します。*str*は、セットされるリソースの値を持つストリングです。(構文はAppendix Hにあるリソース・ファイルと同じです。)

XrtSetReverseConverters()

*XtConvertAndStore()*と使用される XRT/graph の逆変換プログラムを登録します。逆変換プログラムは、アプリケーションに、人間が読める形のリソース値を提供します。

```
void
XrtSetReverseConverters(void)
```

XrtSetValueLabel()

X、Y、Y2 軸の新しい Value ラベルを定義します。

```
void
XrtSetValueLabel(
  Widget       graph,
  XrtAxis      axis,
  XrtValueLabel    *label
  )
```

graph は、グラフのハンドルです。*axis* は、XRT_AXIS_X、XRT_AXIS_Y、XRT_AXIS_Y2 のどれかです。*label* で指されている XrtValueLabel 構造体の *value* と *string* は、この手続きが呼び出される前にセットされている必要があります。この手続きは、グラフに新しい Value ラベルを追加します。XRT/graph は、*label* で指されている XrtValueLabel をコピーします。

XrtSetValues()

グラフのリソースをセットします。

```
void
XrtSetValues(
  Widget  graph,
  ...,
  NULL
  )
```

graph は、グラフのハンドルです。...は、(一つまたは複数の)*resource* と *value* のペアです。*resource* は、(P267 のリストにある、XrtProperty クロス・プラットフォーム名で指定される)リソースです。*value* は、新しい値を指定します。*value* のデータ型は、リソースのデータ型と同じにしてください。float 型のリソースにアクセスする時は、*XrtFloatToArgVal()*を使用しないでください。*resource* と *value* のペアのリストは、NULL で終了します。

XrtTextAreaCreate()

属性がデフォルト値にセットされたテキスト・オブジェクトを作成します。この手続きは、作成されたテキスト・オブジェクトのハンドルを返します。

```
XrtTextHandle
XrtTextAreaCreate(
  Widget  parent
  )
```

XrtTextAreaDestroy()

テキスト・オブジェクトを破棄します。*XrtTextAreaCreate()*で作成されたオブジェクトにのみ使用できます。

```
void
XrtTextAreaDestroy(
  XrtTextHandle    text
  )
```

text は、破棄されるテキスト・オブジェクトのハンドルです。オブジェクトが他のグラフで使用されている場合は、自動的に切り離されます。

XrtTextAreaGetPropString()

逆変換プログラムを使用して、テキスト・オブジェクトの属性の現在値をストリングとして取得します。

```
Boolean
XrtTextAreaGetPropString(
      XrtTextHandle    hText,
      XrtProperty  resource,
      Char     **str /* Returned */
      )
```

hText はテキスト・オブジェクトのハンドル、*resource* は TextArea の属性を指定します。*str* は、返されたリソース値のストリングのポインタです。*str* は使用後、*XrtFreePropString()*によって必ず解放してください。

XrtTextAreaGetValues()

テキスト・オブジェクトの属性の現在値を取得します。

```
void
XrtTextAreaGetValues(
  XrtTextHandle    hText
  ...,
  NULL
  )
```

hText は、テキスト・オブジェクトのハンドルです。...は、(一つまたは複数の)*attribute* と *value* のペアです。*attribute* は、(Appendix B のリストにある)属性です。*value* は、返された属性の値を持つ変数のポインタです。*value* のデータ型は、属性のデータ型と同じにしてください。*attribute* と *value* のペアのリストは、NULL で終了します。

XrtTextAreaSetPropString()

テキスト・オブジェクトの属性をストリングで表される値にセットします。

```
Boolean
XrtTextAreaSetPropString(
      XrtTextHandle    hText,
      XrtProperty  resource,
      char     *str
      )
```

hText はテキスト・オブジェクトのハンドル、*resource* は TextArea の属性を指定します。*str* は、リソースをセットする時に使用するストリングのポインタです。(構文は、リソース・ファイルの構文と同じです。Appendix H 参照)

XrtTextAreaSetValues()

テキスト・オブジェクトの属性をセットします。

```
void
XrtTextAreaSetValues(
  XrtTextHandle    hText
  ...,
  NULL
  )
```

hText は、テキスト・オブジェクトのハンドルです。...は、(一つまたは複数の)*attribute* と *value* のペアです。*attribute* は、(Appendix B のリストにある)属性です。*value* は、新しい値を指定します。*value* のデータ型は、属性のデータ型と同じにしてください。`float` 型の属性にアクセスする時は、*XrtFloatToArgVal()*を使用しないでください。*attribute* と *value* のペアのリストは、`NULL` で終了します。

XrtTextAttach()

グラフにテキスト領域を加えます。(注意) この手続きは、前のバージョンとの互換性を保つためにサポートされています。テキストを貼り付ける時は、テキスト・オブジェクトのAPI(7.3参照)を使用してください。

```
void
XrtTextAttach(
  Widget       graph,
  XrtTextHandle    handle
  )
```

XrtTextCreate()

新しいテキスト領域の作成に使用されます。(注意) この手続きは、前のバージョンとの互換性を保つためにサポートされています。

```
XrtTextHandle
XrtTextCreate(
  Widget       graph,
  XrtTextDesc      *textd
  )
```

XrtTextDestroy()

テキスト領域を破棄します。(注意) この手続きは、前のバージョンとの互換性を保つためにサポートされています。

```
void
XrtTextDestroy(
  Widget       graph,
  XrtTextHandle    handle
  )
```

XrtTextDetach()

グラフからテキスト領域を取り去ります。(注意) この手続きは、前のバージョンとの互換性を保つためにサポートされています。

```
void
XrtTextDetach(
  Widget       graph,
  XrtTextHandle    handle
  )
```

XrtTextDetail()

テキスト領域の情報を取り出します。*handle*が有効なら1を、それ以外は0を返します。(注意) この手続きは、前のバージョンとの互換性を保つためにサポートされています。

```
int
XrtTextDetail(
  Widget       graph,
  XrtTextHandle    handle,
  XrtTextDesc      *textd
  )
```

XrtTextUpdate()

テキスト領域の情報をアップデートします。(注意) この手続きは、前のバージョンとの互換性を保つためにサポートされています。

```
void
XrtTextUpdate(
  Widget       graph,
  XrtTextHandle    handle,
  XrtTextDesc      *textd
  )
```

XrtUnmap()

グラフ座標からpixel座標へアンマップします。

```
void
XrtUnmap(
  Widget       graph,
  int          yaxis,
  double       x,
  doble        y,
  XrtMapResult     *map
  )
```

アプリケーションで、(*graph*で指定されるグラフの)グラフ座標(x, y)に対応するpixel座標を決定します。*yaxis*を1にセットすると一番目のY軸、2にすると二番目のY軸で定義された座標系からアンマップします。結果は、*map*で指されている構造体の*pix_x*と*pix_y*に返されます。範囲外のアンマップのリクエストに対しては、値が-1の*pix_x*と*pix_y*が返されます。

XrtUnpick()

*set*と*point*を与えられたpixel座標を調べます。

```
void
XrtUnpick(
  Widget       graph,
  int          dataset,
  int          set,
  int          point,
  XrtPickResult    *pick
  )
```

アプリケーションで、データが表示されているpixel座標を決定します。

複合グラフの二番目のデータ・セットからアンピックするのでなければ、*dataset*を XRT_DATASET1に設定しておいてください。*set*と*point*は、データ・セットのインデックスです。

結果は、*pick*で指されている構造体の*pix_x*と*pix_y*に返されます。データが範囲外の場合は、*pix_x*と*pick_y*は両方とも-1になります。

XtCreateXrtGraph()

XRT/graph ウィジェットを作成します。

```
Widget
XtCreateXrtGraph(
  Widget  parent,
  char*name,
  Arg     *arglist,
  int     argcount
  )
```

parent は親です。*name* は作成されたウィジェットの名前、*arglist* は *argcount* 引数のリストです。(*XtSetArg()*で作成されます。)

XtIsXrtGraph()

ウィジェットが XRT/graph ウィジェットのサブクラスであれば、True を返します。

```
Boolean
XtIsXrtGraph(
  Widget  graph
  )
```

E

トランスレーションとアクション

デフォルトのトランスレーション
アクション・ルーチン

E.1 デフォルトのトランスレーション

XRT/graphは、トランスレーション(アクションとイベントの定義)をXmPrimitiveから継承し、以下のような独自のトランスレーションを定義をします。

Event	Action
Shift <Btn2Down>	ModifyStart(Translate)
Ctrl <Btn2Down>	ModifyStart(Scale)
~Shift ~Ctrl <Btn2Down>	ModifyStart(Rotate)
Ctrl <Btn1Down>	ModifyStart(Zoom)
Shift <Btn1Down>	ModifyStart(ZoomAxis)
<Key> r	ModifyStart(Reset) ViewReset() ModifyEnd(Reset)
~Shift ~Ctrl <Btn3Down>	Properties()
Shift <Btn2Motion>	TranslateUpdate()
Ctrl <Btn2Motion>	ScaleUpdate()
~Shift ~Ctrl <Btn2Motion>	RotateUpdate()
Ctrl <Btn1Motion>	ZoomUpdate()
Shift <Btn1Motion>	ZoomAxisUpdate()

Event	Action
Shift <Btn2Up>	ModifyEnd(Translate)
Ctrl <Btn2Up>	ModifyEnd(Scale)
~Shift ~Ctrl <Btn2Up>	ModifyEnd(Rotate)
Ctrl <Btn1Up>	ZoomEnd() ModifyEnd(Zoom)
Shift <Btn1Up>	ZoomAxisEnd() ModifyEnd(ZoomAxis)
Ctrl Button1 <Key> osfCancel[a]	ZoomCancel() ModifyEnd(Zoom)
Shift Button1 <Key> osfCancel[a]	ZoomAxisCancel() ModifyEnd(ZoomAxis)
Ctrl <KeyUp>	ZoomCancel()
Shift <KeyUp>	ZoomCancel()

a: XRT/graph ウィジェットが、Form または BulletinBoard ウィジェットの中で作成された場合は、osfCancel イベントを受け取りません。キャンセルのアクションには、別のイベントを定義してください。

E.2 アクション・ルーチン

XRT/graph が定義しているアクション・ルーチンのアルファベット順のリストです。

Map()

ユーザ・イベントの pixel 座標にマップするグラフ座標を決定します。デフォルトでは、このアクションは、どのイベントにも定義されていません。*Map()*は、*XrtMap()*を呼び出し、`XtNxrtMapCallback` で指定されるコールバックを開始し、グラフ座標を持つ `XrtMapCallbackStruct` 構造体のポインタを各コールバックに渡します。*Map()*には、マップする座標の Y 軸を指定するオプションのパラメータが一つあります。

`YAxis1` 一番めの Y 軸で定義される Y 軸座標

`Yaxis2` 二番めの Y 軸で定義される Y 軸座標

パラメータが指定されていない場合は、*Map()*は一番めの Y 軸を使用します。

ModifyStart()

ユーザ・インタラクションを開始し、アプリケーションにそれを不許可にする方法を提供します。*ModifyStart()*は、`XtNxrtModifyCallback` で指定されるコールバックを開始し、*reason* が `XRT_REASON_MODIFY_START` にセットされた `XrtModifyCallbackStruct` 構造体のポインタを各コールバックに渡します。アプリケーションは、*doit* を `False` にセットして、全てのユーザ・インタラクションを不可にすることがきます。

*ModifyStart()*は、行われようとするユーザ・インタラクションを示すパラメータを一つ取ります。

`Translate`	ユーザが行おうとするグラフの移動
`Scale`	ユーザが行おうとするグラフのスケーリング
`Rotate`	ユーザが行おうとするグラフの回転
`Zoom`	ユーザが行おうとするグラフのズーム
`ZoomAxis`	ユーザが行おうとする軸を表示したままのグラフのズーム
`Reset`	ユーザが行おうとするグラフのリセット

ModifyEnd()

ユーザ・インタラクションが終了したことをアプリケーションに伝えます。*ModifyEnd()*は、`XtNxrtModifyCallback` で指定されるコールバックを開始し、*reason* が `XRT_REASON_MODIFY_END` にセットされた `XrtModifyCallbackStruct` 構造体のポインタを各コールバックに渡します。*ModifyEnd()*は、終了したユーザ・インタラクションを示すパラメータを一つ取ります。

`Translate`	インタラクティブな移動の終了
`Scale`	インタラクティブなスケーリングの終了
`Rotate`	インタラクティブな回転の終了
`Zoom`	インタラクティブなズームの終了
`ZoomAxis`	インタラクティブな軸のズームの終了
`Reset`	リセットの終了

NoAction()

何も行われず、コールバックも開始されません。デフォルトでは、このアクションはどのイベントにも定義されていません。このルーチンを呼ぶトランスレーションをセットして、ユーザ・イベントを効率的に不可にすることができます。例えば、スケーリングをオフにするために、`Ctrl<Btn2Motion>`が*NoAction()*を呼ぶように定義することができます。

Pick()

ユーザ・イベントの pixel 座標に一番近いグラフのデータ値を決定します。デフォルトでは、このアクションはどのイベントにも定義されていません。*Pick()*は、*XrtPick()*を呼び出し、`XtNxrtPickCallback` で指定されるコールバックを開始、グラフのデータ値を持つ `XrtPickCallbackStruct` 構造体のポインタを各コールバックに返します。

*Pick()*は、データセットと対象となる軸を指定する、二つのオプションのパラメータ(Dataset、AxisFocus)を取ります。

Dataset	Data1 は一番めのデータセット、Data2 は二番めのデータセット、AllData は両方のデータセットを使用します。
AxisFocus	XFocus は、一番近いデータ・ポイントの検索に X 軸を使用し、YFocus は Y 軸、XYFocus は両方の軸を使用します。

パラメータが指定されない場合は、*Pick()*は(AllData、XYFocus)を使用します。

Properties()

XRT/graph のプロパティ・エディタ(Property Editor)を実行し、ユーザがリソース値を編集できるようにします。プロパティ・エディタは、*XrtPropertyPageEnable()*を使用して、使用可能に設定されている場合にのみ表示されます。コールバックは開始されません。

RotateUpdate()

棒と円グラフを(XtNxrtGraphDepth が>0 であれば)回転します。*RotateUpdate()*は、XtNxrtGraphInclination と XtNxrtGraphRotation リソースの新しい値を計算し、次に、XtNxrtRotateCallback で指定されるコールバックを開始し、XrtRotateCallbackStruct 構造体のポインタを各コールバックに渡します。アプリケーションが XrtRotateCallbackStruct の *doit* パラメータを False にしていなければ、*RotateUpdate()*は新しい回転の値をウィジェットに適用します。(XtNxrtExposeCallback を開始します。)

ScaleUpdate()

グラフのスケーリングを行います。*ScaleUpdate()*は、グラフのマージンの新しい値を計算し、次に、XtNxrtTransformCallback で指定されたコールバックを開始し、XrtTransformCallbackStruct 構造体のポインタを各コールバックに渡します。アプリケーションが XrtTransformCallbackStruct の *doit* パラメータを False にしていなければ、*ScaleUpdate()* は新しいグラフのマージンをウィジェットに適用します。(XtNxrtExposeCallback を開始します。)

TranslateUpdate()

グラフを移動します。*TranslateUpdate()*はグラフのマージンの新しい値を計算し、次に XtNxrtTransformCallback で指定されたコールバックを開始し、XrtTransformCallbackStruct 構造体のポインタを各コールバックに渡します。アプリケーションが XrtTransformCallbackStruct の *doit* パラメータを False にしていなければ、*TranslateUpdate()* は新しいグラフのマージンをウィジェットに適用します。(XtNxrtExposeCallback を開始します。)

ViewReset()

グラフのスケーリング、移動、ズームを XRT/graph のデフォルトに戻します。ウィジェットに対するユーザ・インタラクションがあった場合にのみ実行されます。*ViewReset()*は、XtNxrtTransformCallback で指定されたコールバックを開始し、次に、デフォルトのグラフ・マージン値を持ち、*reset* パラメータが True にセットされた XrtTransformCallbackStruct 構造体のポインタを各コールバックに渡します。

*ViewReset()*は、その後、XtNxrtZoomAxisCallback で指定されたコールバックを開始し、デフォルトの軸の範囲を持ち、*reset* パラメータが True にセットされた XrtZoomAxisCallbackStruct 構造体のポインタを各コールバックに渡します。アプリケーションが XrtTransformCallbackStruct の *doit* パラメータを False にしていなければ、*ViewReset()*は新しいグラフのマージンと軸の最小/最大値をウィジェットに適用します。(XtNxrtExposeCallback を開始します。)

ZoomAxisUpdate()
ZoomUpdate()

ズームする領域の“rubberband“をアップデートします。コールバックは開始されません。

ZoomAxisEnd()

軸を表示したまま、“zoom rubberband”で選択された領域をズームします。*ZoomAxisEnd()*は、軸の最小と最大値を計算し、次に、XtNxrtZoomAxisCallback で指定されたコールバックを開始し、XrtZoomAxisCallbackStruct 構造体のポインタを各コールバックに渡します。アプリケーションが XrtZoomAxisCallbackStruct の *doit* パラメータを False にしていなければ、*ZoomAxisEnd()* は、新しい軸の範囲をウィジェットに適用します。(XtNxrtExposeCallback を開始します。)

ZoomEnd()

“zoom rubberband“で選択されている領域をズームします。*ZoomEnd()*は、新しいグラフのマージンの値を計算し、次に XtNxrtTransformCallback で指定されるコールバックを開始し、XrtTransformCallbackStruct 構造体のポインタを各コールバックに渡します。アプリケーションが XrtTransformCallbackStruct の *doit* パラメータを False にしていなければ、*ZoomEnd()* は新しいグラフのマージンをウィジェットに適用します。(XtNxrtExposeCallback を開始します。)

ZoomAxisCancel()
ZoomCancel()

進行中のズームをキャンセルし、“zoom rubberband“を消去します。

F

データ型

XRT/graphのデータ型のアルファベット順のリストです。構造体のCの定義も提供してあります。

XrtAdjust

XtNxrtFooterAdjustとXtNxrtHeaderAdjustのリソースに使用される列挙法。

```
XRT_ADJUST_LEFT
XRT_ADJUST_RIGHT
XRT_ADJUST_CENTER
```

XrtAlign

XrtNxrtLegendOrientationリソースに使用される列挙法。

```
XRT_ALIGN_VERTICAL
XRT_ALIGN_HORIZONTAL
```

XrtAnchor

XtNxrtLegendAnchorリソースとテキスト領域に使用される列挙法。

```
XRT_ANCHOR_NORTH
XRT_ANCHOR_SOUTH
XRT_ANCHOR_EAST
XRT_ANCHOR_WEST
XRT_ANCHOR_NORTHWEST
XRT_ANCHOR_NORTHEAST
XRT_ANCHOR_SOUTHWEST
XRT_ANCHOR_SOUTHEAST
XRT_ANCHOR_HOME
XRT_ANCHOR_BEST
```

XRT_ANCHOR_HOMEとXRT_ANCHOR_BESTはテキスト領域にのみ使用されます。

XrtAngleUnit()

XtNxrtAngleUnitリソースに使用される列挙法。

```
XRT_ANGLE_DEGREES
XRT_ANGLE_RADIANS
XRT_ANGLE_GRADS
```

XrtAnnoMethod

X、Y、Y2 軸の注釈法の定義に使用される列挙法。
XtNxrt[XYY2]AnnotationMethod リソースと共に使用されます。

```
XRT_ANNO_VALUES
XRT_ANNO_POINT_LABELS
XRT_ANNO_VALUE_LABELS
XRT_ANNO_TIME_LABELS
```

XrtAnnoPlacement

XtNxrt[XY]AnnoPlacement と共に使用され、軸の注釈の位置を選択する列挙法。

```
XRT_ANNO_AUTO
XRT_ANNO_ORIGIN
XRT_ANNO_MIN
XRT_ANNO_MAX
```

XrtAttachType

テキスト領域の貼り付け方法を指定する列挙法。

```
XRT_TEXT_ATTACH_PIXEL
XRT_TEXT_ATTACH_VALUE
XRT_TEXT_ATTACH_DATA
XRT_TEXT_ATTACH_DATA_VALUE
```

XrtAxis

軸の指定に使用される列挙法。*XrtGetValueLabel()*、*XrtSetValueLabel()*と共に使用されます。

```
XRT_AXIS_X
XRT_AXIS_Y
XRT_AXIS_Y2
```

XrtBorder

XtNxrtBorder 、 XtNxrtHeaderBorder 、 XtNxrtFooterBorder 、 XtNxrtLegendBorder、XtNxrtGraphBorderとテキスト領域に使用される列挙法。

```
XRT_BORDER_NONE
XRT_BORDER_PLAIN
XRT_BORDER_3D_OUT
XRT_BORDER_3D_IN
XRT_BORDER_ETCHED_IN
XRT_BORDER_ETCHED_OUT
XRT_BORDER_FRAME_IN
XRT_BORDER_FRAME_OUT
XRT_BORDER_BEVEL
XRT_BORDER_SHADOW
```

XrtCallbackStruct

XtNxrtExposeCallback と XtNxrtResizeCallback リストのコールバックに渡される情報を定義する構造体。

```
typedef struct {
  int          reason;
  XEvent       *event;
  Window       window;
  Region       region;
  int          change;
} XrtCallbackStruct;
```

XrtColor

*char**として定義される、プラットフォーム固有の型。

XrtDataStyle

データ・セットがグラフでどのように表示されるかを定義する構造体。

```
typedef struct {
  XrtLinePattern   lpat;    /* line pattern */
  XrtFillPattern   fpat;    /* fill pattern */
  XrtColor     color;   /* line color */
  int          width;   /* line width */
  XrtPoint     point;   /* point style */
  XrtColor     pcolor;  /* point color */
  int          psize;   /* point size */
  unsigned long    res1;    /* reserved */
  unsigned long    res2;    /* reserved */
  unsigned long    resh;    /* reserved */
  unsigned long    ress;    /* reserved */
} XrtDataStyle;
```

XrtDisplay

データの個別のセットをどのように表示するかを指定する列挙法。

```
XRT_DISPLAY_SHOW
XRT_DISPLAY_HIDE
XRT_DISPLAY_EXCLUDE
```

XrtDrawPSColor

*XrtDrawPS()*で色の処理を指定する列挙法。

```
XRT_PS_COLOR_AS_IS
XRT_PS_COLOR_AUTO
XRT_PS_COLOR_MONO
```

XrtFillPattern

XrtDataStyle で使用される塗りつぶしパターンの列挙法。

```
XRT_FPAT_NONE
XRT_FPAT_SOLID
XRT_FPAT_25_PERCENT
XRT_FPAT_50_PERCENT
XRT_FPAT_75_PERCENT
XRT_FPAT_HORIZ_STRIPE
XRT_FPAT_VERT_STRIPE
XRT_FPAT_45_STRIPE
XRT_FPAT_135_STRIPE
XRT_FPAT_DIAG_HATCHED
XRT_FPAT_CROSS_HATCHED
```

XrtImage

XtPointer として定義される、プラットフォーム固有の型。

注意: この型は、(同じく XtPointer として定義される)XrtPDSImage と一緒に使用できます。

XrtLinePattern

XrtDataStyleで使用される線のパターンの列挙法。

```
XRT_LPAT_NONE
XRT_LPAT_SOLID
XRT_LPAT_LONG_DASH
XRT_LPAT_DOTTED
XRT_LPAT_SHORT_DASH
XRT_LPAT_LSL_DASH
XRT_LPAT_DASH_DOT
```

XrtMapCallbackStruct

XtNxrtMapCallback リストのコールバックに渡される情報を定義する構造体。

```
typedef struct {
  int          reason;
  XEvent       *event;
  String       *params;
  Cardinal     nparams;
  XrtRegion        region;
  XrtMapResult     map;
} XrtMapCallbackStruct;
```

XrtMapResult

マップされたpixel座標の情報を渡す時に使用される構造体です。

```
typedef struct {
  int      pix_x, pix_y;
  int      yaxis;
  float        x, y;
} XrtMapResult;
```

XrtMarkerMethod

XtNxrtXMarkerMethod に使用され、X マーカの位置決め法を選択する列挙法。

```
XRT_MARKER_SET_POINT
XRT_MARKER_VALUE
```

XrtModifyCallbackStruct

XtNxrtModifyCallback リストのコールバックに渡される情報を定義する構造体。

```
typedef struct {
  int          reason;
  XEvent       *event;
  String       *params;
  Cardinal     nparams;
  Boolean      doit;
} XrtModifyCallbackStruct;
```

XrtNumMethod

XtNxrt[XYY2]NumMethod に使用され、軸の注釈の計算方法を指定する列挙法。

```
XRT_NUM_PRECISION
XRT_NUM_ROUND
```

XrtOriginPlacement

デフォルトの原点の選択に使用される列挙法。

```
XRT_ORIGIN_AUTO
XRT_ORIGIN_ZERO
XRT_ORIGIN_MIN
XRT_ORIGIN_MAX
```

XrtPickCallbackStruct

XtNxrtPickCallbackリストのコールバックに渡される情報を定義する構造体。

```
typedef struct {
  int          reason;
  XEvent       *event;
  String       *params;
  Cardinal     nparams;
  XrtRegion        region;
  XrtPickResult    pick;
} XrtPickCallbackStruct;
```

XrtPickResult

ピックされたpixel座標の情報を渡す時に使用される構造体。

```
typedef struct {
  int      pix_x, pix_y;
  int      dataset;
  int      set, point;
  int      distance;
} XrtPickResult;
```

XrtPieOrder

円グラフのスライスの順序の列挙法。

```
XRT_PIEORDER_ASCENDING
XRT_PIEORDER_DESCENDIN
XRT_PIEORDER_DATA_ORDER
```

XrtPieThresholdMethod

円グラフのしきい値法の列挙法。

```
XRT_PIE_SLICE_CUTOFF
XRT_PIE_PERCENTILE
```

XrtPoint

XrtDataStyleで使用されるポイント・スタイルの列挙法。

```
XRT_POINT_NONE
XRT_POINT_DOT
XRT_POINT_BOX
XRT_POINT_TRI
XRT_POINT_DIAMOND
XRT_POINT_STAR
XRT_POINT_VERT_LINE
XRT_POINT_HORIZ_LINE
XRT_POINT_CROSS
XRT_POINT_CIRCLE
XRT_POINT_SQUARE
XRT_POINT_INVERT_TRI
XRT_POINT_DIAG_CROSS
XRT_POINT_OPEN_TRI
XRT_POINT_OPEN_DIAMOND
XRT_POINT_OPEN_INVERT_TRI
```

XrtProperty

XrtGetPropString() 、 *XrtGetValues()* 、 *XrtSetPropString()* 、 *XrtTextAreaSetPropString()* 、 *XrtTextAreaGetPropString()*、*XrtSetValues()*に使用される、XRT/graph リソースのクロス・プラットフォーム名の列挙法。(各リソースの定義は、P267 を参照してください。)

XrtReason

コールバックの理由 (reason) を定義する列挙法。XtNxrtMapCallback 、XtNxrtModifyCallback 、 XtNxrtPickCallback 、 XtNxrtRotateCallback 、XtNxrtTransformCallback、XtNxrtZoomAxisCallback に使用されます。

```
XRT_REASON_MODIFY_START
XRT_REASON_MODIFY_END
XRT_REASON_ROTATE
XRT_REASON_TRANSFORM
XRT_REASON_ZOOM_AXIS
XRT_REASON_MAP
XRT_REASON_PICK
```

XrtRegion

マップとピックの結果の列挙法です。*XrtMap()*と*XrtPick()*によって返されます。

```
XRT_RGN_NOWHERE
XRT_RGN_IN_GRAPH
XRT_RGN_IN_LEGEND
XRT_RGN_IN_FOOTER
XRT-RGN_IN_HEADER
```

XrtRotate

反時計方向の回転の角度の定義に使用される列挙法。タイトルと注釈の回転のリソースとテキスト・オブジェクトに使用されます。

```
XRT_ROTATE_NONE
XRT_ROTATE_90
XRT_ROTATE_270
```

XrtRotateCallbackStruct

XtNxrtRotateCallback リストのコールバックに渡される情報を定義する構造体。

```
typedef struct {
  int          reason;  // Read-only
  XEvent       *event;  // Read-only
  String       *params; // Read-only
  Cardinal     nparams; // Read-only
  int          rotation;
  int          inclination;
  Boolean      doit
} XrtRotateCallbackStruct;
```

XrtShading

3D 表示されているグラフの影をコントロールする列挙法。

```
XRT_SHADING_COLOR
XRT_SHADING_DITHERED
XRT_SHADING_NONE
```

XrtTimeUnit

時間の単位の定義に使用される列挙法。XtNxrtTimeUnitに使用されます。

```
XRT_TMUNIT_SECONDS
XRT_TMUNIT_MINUTES
XRT_TMUNIT_HOURS
XRT_TMUNIT_DAYS
XRT_TMUNIT_WEEKS
XRT_TMUNIT_MONTHS
XRT_TMUNIT_YEARS
```

XrtTransformCallbackStruct

XtNxrtTransformCallbackリストのコールバックに渡される情報を定義する構造体。

```
typedef struct {
  int          reason;  // Read-only
  XEvent       *event;  // Read-only
  String       *params; // Read-only
  Cardinal     nparams; // Read-only
  Position     left_margin;
  Position     right_margin;
  Position     top_margin;
  Position     bottom_margin;
  Boolean      doit;
} XrtTransformCallbackStruct;
```

XrtType

グラフ・タイプの定義に使用される列挙法。XtNxrtTypeに使用されます。

```
XRT_TYPE_PLOT
XRT_TYPE_BAR
XRT_TYPE_PIE
XRT_TYPE_STACKING_BAR
XRT_TYPE_AREA
XRT_TYPE_HILO
XRT_TYPE_HILO_OPEN_CLOSE
XRT_TYPE_CANDLE
XRT_TYPE_BUBBLE
XRT_TYPE_POLAR
XRT_TYPE_RADAR
XRT_TYPE_FILLED_RADAR
```

XrtValueLabel

X、Y、Y2 軸の一つの値ラベル(Value ラベル)を定義する構造体。XtNxrt[XYY2]Labelsに使用されます。

```
typedef struct {
  float   value;
  char*string;
} XrtValueLabel;
```

XrtZoomAxisCallbackStruct

XtNxrtZoomAxisCallback リストのコールバックに渡される情報を定義する構造体。

```
typedef struct {
  int          reason;  // Read-only
  XEvent       *event;  // Read-only
  String       *params; // Read-only
  Cardinal     nparams; // Read-only
  Boolean      reset;   // Read-only
  float        new_xaxis_min;
  float        new_xaxis_max;
  float        new_yaxis_min;
  float        new_yaxis_max;
  float        new_y2axis_min;
  float        new_y2axis_max;
  Boolean      doit;
} XrtZoomAxisCallbackStruct;
```

G

リソース名、型とクラス

リソースの型とクラスのアルファベット順のリストです。各リソース名の下には、XrtProperty の列挙法が記載されています。

Name	Class	Type
XtNxrtAngleUnit XRT_ANGLE_UNIT	XtCXrtAngleUnit	XtRXrtAngleUnit
XtNxrtAxisBoundingBox XRT_AXIS_BOUNDING_BOX	XtCXrtDrawOption	XtRBoolean
XtNxrtAxisFont XRT_AXIS_FONT	XmCFontList	XmRFontList
XtNxrtBackgroundColor XRT_BACKGROUND_COLOR	XtCXrtBackgroundColor	XtRXrtString
XtNxrtBarClusterOverlap XRT_BAR_CLUSTER_OVERLAP	XtCXrtBarClusterOverlap	XtRInt
XtNxrtBarClusterWidth XRT_BAR_CLUSTER_WIDTH	XtCXrtBarClusterWidth	XtRInt
XtNxrtBorder XRT_BORDER	XtCXrtBorder	XtRXrtBorder
XtNxrtBubbleMax XRT_BUBBLE_MAX	XtCXrtTypeBubbleBounds	XtRFloat
XtNxrtBubbleMethod XRT_BUBBLE_METHOD	XtCXrtBubbleMethod	XtRXrtBubbleMethod
XtNxrtBubbleMin XRT_BUBBLE_MIN	XtCXrtTypeBubbleBounds	XtRFloat
XtNxrtCandleComplex XRT_CANDLE_COMPLEX	XtCXrtCandleOption	XtRBoolean
XtNxrtColorFunc XRT_COLOR_FUNC	XtCXrtFunction	XtRFunction

Name	Class	Type
XtNxrtData XRT_DATA	XtCXrtData	XtRXrtData
XtNxrtData2 XRT_DATA2	XtCXrtData	XtRXrtData
XtNxrtDataAreaBackgroundColor XRT_DATA_AREA_BACKGROUND_COLOR	XtCXrtBackgroundColor	XtRXrtColor
XtNxrtDataAreaForegroundColor XRT_DATA_AREA_FOREGROUND_COLOR	XtCXrtForegroundColor	XtRXrtColor
XtNxrtDataAreaImage XRT_DATA_AREA_IMAGE	XtCXrtImage	XtRXrtImage
XtNxrtDataAreaImageLayout XRT_DATA_AREA_IMAGE_LAYOUT	XtCXrtImageLayout	XtRXrtImageLayout
XtNxrtDataStyles XRT_DATA_STYLES	XtCXrtDataStyles	XtRXrtDataStyles
XtNxrtDataStyles2 XRT_DATA_STYLES2	XtCXrtDataStyles	XtRXrtDataStyles
XtNxrtDataStylesUseDefault XRT_DATA_STYLES_USE_DEFAULT	XtCXrtDataStylesUseDefault	XtRBoolean
XtNxrtDataStyles2UseDefault XRT_DATA_STYLES2_USE_DEFAULT	XtCXrtDataStylesUseDefault	XtRBoolean
XtNxrtDebug XRT_DEBUG	XtCXrtDebug	XtRBoolean
XtNxrtDoubleBuffer XRT_DOUBLE_BUFFER	XtCXrtDoubleBuffer	XtRBoolean
XtNxrtExposeCallback[a]	XtCCallback	XtRCallback
XtNxrtExtraDefaultDataStyles XRT_EXTRA_DEFAULT_DATA_STYLES	XtCXrtDataStylesUseDefault	XtRBoolean
XtNxrtFooterAdjust XRT_FOOTER_ADJUST	XtCXrtAdjust	XtRXrtAdjust
XtNxrtFooterBackgroundColor XRT_FOOTER_BACKGROUND_COLOR	XtCXrtBackgroundColor	XtRXrtColor
XtNxrtFooterBorder XRT_FOOTER_BORDER	XtCXrtBorder	XtRXrtBorder
XtNxrtFooterBorderWidth XRT_FOOTER_BORDER_WIDTH	XtCXrtBorderWidth	XtRDimension
XtNxrtFooterFont XRT_FOOTER_FONT	XmCFontList	XmRFontList
XtNxrtFooterForegroundColor XRT_FOOTER_FOREGROUND_COLOR	XtCXrtForegroundColor	XtRXrtColor
XtNxrtFooterHeight XRT_FOOTER_HEIGHT	XtCXrtDimension	XtRShort

Name	Class	Type
XtNxrtFooterImage XRT_FOOTER_IMAGE	XtCXrtImage	XtRXrtImage
XtNxrtFooterImageLayout XRT_FOOTER_IMAGE_LAYOUT	XtCXrtImageLayout	XtRXrtImageLayout
XtNxrtFooterImageMinimumSize XRT_FOOTER_IMAGE_MINIMUM_SIZE	XtCXrtImageMinimumSize	XtRXBoolean
XtNxrtFooterStrings XRT_FOOTER_STRINGS	XtCXrtStrings	XtRXrtStrings
XtNxrtFooterWidth XRT_FOOTER_WIDTH	XtCXrtDimension	XtRShort
XtNxrtFooterX XRT_FOOTER_X	XtCXrtPosition	XtRShort
XtNxrtFooterXUseDefault XRT_FOOTER_X_USE_DEFAULT	XtCXrtPositionUseDefault	XtRBoolean
XtNxrtFooterY XRT_FOOTER_Y	XtCXrtPosition	XtRShort
XtNxrtFooterYUseDefault XRT_FOOTER_Y_USE_DEFAULT	XtCXrtPositionUseDefault	XtRBoolean
XtNxrtForegroundColor XRT_FOREGROUND_COLOR	XtCXrtForegroundColor	XtRXrtColor
XtNxrtFrontDataset XRT_FRONT_DATASET	XtCXrtDataset	XtRInt
XtNxrtGraph3dShading XRT_GRAPH_3D_SHADING	XtCXrtGraph3dShading	XtRXrtShading
XtNxrtGraphBackgroundColor XRT_GRAPH_BACKGROUND_COLOR	XtCXrtBackgroundColor	XtRXrtColor
XtNxrtGraphBorder XRT_GRAPH_BORDER	XtCXrtBorder	XtRXrtBorder
XtNxrtGraphBorderWidth XRT_GRAPH_BORDER_WIDTH	XtCXrtBorderWidth	XtRDimension
XtNxrtGraphDepth XRT_GRAPH_DEPTH	XtCXrtGraphDepth	XtRInt
XtNxrtGraphForegroundColor XRT_GRAPH_FOREGROUND_COLOR	XtCXrtForegroundColor	XtRXrtColor
XtNxrtGraphHeight XRT_GRAPH_HEIGHT	XtCXrtDimension	XtRShort
XtNxrtGraphHeightUseDefault XRT_GRAPH_HEIGHT_USE_DEFAULT	XtCXrtDimensionUseDefault	XtRBoolean
XtNxrtGraphImage XRT_GRAPH_IMAGE	XtCXrtImage	XtRXrtImage

Name	Class	Type
XtNxrtGraphImageLayout XRT_GRAPH_IMAGE_LAYOUT	XtCXrtImageLayout	XtRXrtImageLayout
XtNxrtGraphInclination XRT_GRAPH_INCLINATION	XtCXrtGraphInclination	XtRInt
XtNxrtGraphMarginBottom XRT_GRAPH_MARGIN_BOTTOM	XtCXrtGraphMargin	XtRShort
XtNxrtGraphMarginBottomUseDefault XRT_GRAPH_MARGIN_BOTTOM_USE_DEFAULT	XtCXrtGraphMarginUseDefault	XtRBoolean
XtNxrtGraphMarginLeft XRT_GRAPH_MARGIN_LEFT	XtCXrtGraphMargin	XtRShort
XtNxrtGraphMarginLeftUseDefault XRT_GRAPH_MARGIN_LEFT_USE_DEFAULT	XtCXrtGraphMarginUseDefault	XtRBoolean
XtNxrtGraphMarginRight XRT_GRAPH_MARGIN_RIGHT	XtCXrtGraphMargin	XtRShort
XtNxrtGraphMarginRightUseDefault XRT_GRAPH_MARGIN_RIGHT_USE_DEFAULT	XtCXrtGraphMarginUseDefault	XtRBoolean
XtNxrtGraphMarginTop XRT_GRAPH_MARGIN_TOP	XtCXrtGraphMargin	XtRShort
XtNxrtGraphMarginTopUseDefault XRT_GRAPH_MARGIN_TOP_USE_DEFAULT	XtCXrtGraphMarginUseDefault	XtRBoolean
XtNxrtGraphRotation XRT_GRAPH_ROTATION	XtCXrtGraphRotation	XtRInt
XtNxrtGraphShowOutlines XRT_GRAPH_SHOW_OUTLINES	XtCXrtGraphOption	XtRBoolean
XtNxrtGraphWidth XRT_GRAPH_WIDTH	XtCXrtDimension	XtRShort
XtNxrtGraphWidthUseDefault XRT_GRAPH_WIDTH_USE_DEFAULT	XtCXrtDimensionUseDefault	XtRBoolean
XtNxrtGraphX XRT_GRAPH_X	XtCXrtPosition	XtRShort
XtNxrtGraphXUseDefault XRT_GRAPH_X_USE_DEFAULT	XtCXrtPositionUseDefault	XtRBoolean
XtNxrtGraphY XRT_GRAPH_Y	XtCXrtPosition	XtRShort
XtNxrtGraphYUseDefault XRT_GRAPH_Y_USE_DEFAULT	XtCXrtPositionUseDefault	XtRBoolean
XtNxrtHeaderAdjust XRT_HEADER_ADJUST	XtCXrtAdjust	XtRXrtAdjust
XtNxrtHeaderBackgroundColor XRT_HEADER_BACKGROUND_COLOR	XtCXrtBackgroundColor	XtRXrtColor

Name	Class	Type
XtNxrtHeaderBorder XRT_HEADER_BORDER	XtCXrtBorder	XtRXrtBorder
XtNxrtHeaderBorderWidth XRT_HEADER_BORDER_WIDTH	XtCXrtBorderWidth	XtRDimension
XtNxrtHeaderFont XRT_HEADER_FONT	XmCFontList	XmRFontList
XtNxrtHeaderForegroundColor XRT_HEADER_FOREGROUND_COLOR	XtCXrtForegroundColor	XtRXrtColor
XtNxrtHeaderHeight XRT_HEADER_HEIGHT	XtCXrtDimension	XtRShort
XtNxrtHeaderImage XRT_HEADER_IMAGE	XtCXrtImage	XtRXrtImage
XtNxrtHeaderImageLayout XRT_HEADER_IMAGE_LAYOUT	XtCXrtImageLayout	XtRXrtImageLayout
XtNxrtHeaderImageMinimumSize XRT_HEADER_IMAGE_MINIMUM_SIZE	XtCXrtImageMinimumSize	XtRBoolean
XtNxrtHeaderStrings XRT_HEADER_STRINGS	XtCXrtStrings	XtRXrtStrings
XtNxrtHeaderWidth XRT_HEADER_WIDTH	XtCXrtDimension	XtRShort
XtNxrtHeaderX XRT_HEADER_X	XtCXrtPosition	XtRShort
XtNxrtHeaderXUseDefault XRT_HEADER_X_USE_DEFAULT	XtCXrtPositionUseDefault	XtRBoolean
XtNxrtHeaderY XRT_HEADER_Y	XtCXrtPosition	XtRShort
XtNxrtHeaderYUseDefault XRT_HEADER_Y_USE_DEFAULT	XtCXrtPositionUseDefault	XtRBoolean
XtNxrtHiLoCloseShow XRT_HILO_CLOSE_SHOW	XtCXrtHiLoOption	XtRBoolean
XtNxrtHiLoOpenCloseFulWidth XRT_HILO_OPEN_CLOSE_FULL_WIDTH	XtCXrtHiLoOption	XtRBoolean
XtNxrtHiLoOpenShow XRT_HILO_OPEN_SHOW	XtCXrtHiLoOption	XtRBoolean
XtNxrtImage XRT_IMAGE	XtCXrtImage	XtRXrtImage
XtNxrtImageLayout XRT_IMAGE_LAYOUT	XtCXrtImageLayout	XtRXrtImageLayout
XtNxrtInvertOrientation XRT_INVERT_ORIENTATION	XtCXrtDrawOption	XtRBoolean

Name	Class	Type
XtNxrtIsStacked XRT_IS_STACKED	XtCXrtIsStacked	XtRBoolean
XtNxrtIsStacked2 XRT_IS_STACKED2	XtCXrtIsStacked	XtCXrtIsStacked
XtNxrtLegendAnchor XRT_LEGEND_ANCHOR	XtCXrtAnchor	XtRXrtAnchor
XtNxrtLegendBackgroundColor XRT_LEGEND_BACKGROUND_COLOR	XtCXrtBackgroundColor	XtRXrtColor
XtNxrtLegendBorder XRT_LEGEND_BORDER	XtCXrtBorder	XtRXrtBorder
XtNxrtLegendBorderWidth XRT_LEGEND_BORDER_WIDTH	XtCXrtBorderWidth	XtRDimension
XtNxrtLegendFont XRT_LEGEND_FONT	XmCFontList	XmRFontList
XtNxrtLegendForegroundColor XRT_LEGEND_FOREGROUND_COLOR	XtCXrtForegroundColor	XtRXrtColor
XtNxrtLegendHeight XRT_LEGEND_HEIGHT	XtCXrtDimension	XtRShort
XtNxrtLegendImage XRT_LEGEND_IMAGE	XtCXrtImage	XtRXrtImage
XtNxrtLegendImageLayout XRT_LEGEND_IMAGE_LAYOUT	XtCXrtImageLayout	XtRXrtImageLayout
XtNxrtLegendOrientation XRT_LEGEND_ORIENTATION	XtCXrtAlign	XtRXrtAlign
XtNxrtLegendReversed XRT_LEGEND_REVERSED	XtCXrtDrawOption	XtRBoolean
XtNxrtLegendShow XRT_LEGEND_SHOW	XtCXrtDrawOption	XtRBoolean
XtNxrtLegendTitle XRT_LEGEND_TITLE	XtCXrtTitle	XtRXrtString
XtNxrtLegendWidth XRT_LEGEND_WIDTH	XtCXrtDimension	XtRShort
XtNxrtLegendX XRT_LEGEND_X	XtCXrtPosition	XtRShort
XtNxrtLegendXUseDefault XRT_LEGEND_X_USE_DEFAULT	XtCXrtPositionUseDefault	XtRBoolean
XtNxrtLegendY XRT_LEGEND_Y	XtCXrtPosition	XtRShort
XtNxrtLegendYUseDefault XRT_LEGEND_Y_USE_DEFAULT	XtCXrtPositionUseDefault	XtRBoolean
XtNxrtMapCallback[a]	XtCCallback	XtRCallback

Name	Class	Type
XtNxrtMarkerDataset XRT_MARKER_DATASET	XtCXrtDataset	XtRInt
XtNxrtMarkerDataStyle XRT_MARKER_DATA_STYLE	XtCXrtDataStyle	XtRXrtDataStyle
XtNxrtMarkerDataStyleUseDefault XRT_MARKER_DATA_STYLE_USE_DEFAULT	XtCXrtDataStylesUseDefault	XtRBoolean
XtNxrtModifyCallback[a]	XtCCallback	XtRCallback
XtNxrtOtherDataStyle XRT_OTHER_DATA_STYLE	XtCXrtDataStyle	XtRXrtDataStyle
XtNxrtOtherDataStyleUseDefault XRT_OTHER_DATA_STYLE_USE_DEFAULT	XtCXrtDataStylesUseDefault	XtRBoolean
XtNxrtOtherLabel XRT_OTHER_LABEL	XtCXrtOtherLabel	XtRXrtString
XtNxrtPickCallback[a]	XtCCallback	XtRCallback
XtNxrtPieMergeMissingSlices XRT_PIE_MERGE_MISSING_SLICES	XtCXrtPieSlices	XtRBoolean
XtNxrtPieMinSlices XRT_PIE_MIN_SLICES	XtCXrtPieMinSlices	XtRInt
XtNxrtPieOrder XRT_PIE_ORDER	XtCXrtPieOrder	XtRXrtPieOrder
XtNxrtPieStartAngle XRT_PIE_START_ANGLE	XtCXrtPieStartAngle	XtRFloat
XtNxrtPieThresholdMethod XRT_PIE_THRESHOLD_METHOD	XtCXrtPieThresholdMethod	XtRXrtPieThresholdMethod
XtNxrtPieThresholdValue XRT_PIE_THRESHOLD_VALUE	XtCXrtPieThresholdValue	XtRFloat
XtNxrtPointLabels XRT_POINT_LABELS	XtCXrtPointLabels	XtRXrtStrings
XtNxrtPointLabels2 XRT_POINT_LABELS2	XtCXrtPointLabels	XtRXrtStrings
XtNxrtPolarAxisAllowNegatives XRT_POLAR_AXIS_ALLOW_NEGATIVES	XtCXrtPolarOption	XtRBoolean
XtNxrtPolarHalfRange XRT_POLAR_HALF_RANGE	XtCXrtPolarOption	XtRBoolean
XtNxrtRepaint XRT_REPAINT	XtCXrtRepaint	XtRBoolean
XtNxrtResizeCallback[a]	XtCCallback	XtRCallback
XtNxrtRotateCallback[a]	XtCCallback	XtRCallback
XtNxrtSetLabels XRT_SET_LABELS	XtCXrtSetLabels	XtRXrtStrings

Name	Class	Type
XtNxrtSetLabels2 XRT_SET_LABELS2	XtCXrtSetLabels	XtRXrtStrings
XtNxrtTimeBase XRT_TIME_BASE	XtCXrtTimeBase	XtRLong
XtNxrtTimeFormat XRT_TIME_FORMAT	XtCXrtTimeFormat	XtRXrtString
XtNxrtTimeFormatUseDefault XRT_TIME_FORMAT_USE_DEFAULT	XtCXrtTimeFormatUseDefault	XtRBoolean
XtNxrtTimeUnit XRT_TIME_UNIT	XtCXrtTimeUnit	XtRXrtTimeUnit
XtNxrtTransformCallback[a]	XtCCallback	XtRCallback
XtNxrtType XRT_TYPE	XtCXrtType	XtRXrtType
XtNxrtType2 XRT_TYPE2	XtCXrtType	XtRXrtType
XtNxrt[XYY2]AnnoPlacement XRT_[XYY2]ANNO_PLACEMENT	XtCXrtAnnoPlacement	XtRXrtAnnoPlacement
XtNxrt[XYY2]AnnotationMethod XRT_[XYY2]ANNOTATION_METHOD	XtCXrtAnnotationMethod	XtRXrtAnnotationMethod
XtNxrt[XYY2]AnnotationRotation XRT_[XYY2]ANNOTATION_ROTATION	XtCXrtRotation	XtRXrtRotate
XtNxrt[XYY2]AxisDataStyle XRT_[XYY2]AXIS_DATA_STYLE	XtCXrtDataStyle	XtRXrtDataStyle
XtNxrt[XYY2]AxisDataStyleUseDefault XRT_[XYY2]AXIS_DATA_STYLE_USE_DEFAULT	XtCXrtDataStylesUseDefault	XtRBoolean
XtNxrt[XYY2]AxisLabelFormat XRT_[XYY2]AXIS_LABEL_FORMAT	XtCXrtAxisLabelFormat	XtRXrtString
XtNxrt[XYY2]AxisLogarithmic XRT_[XYY2]AXIS_LOGARITHMIC	XtCXrtAxisLogarithmic	XtRBoolean
XtNxrt[XYY2]AxisMax XRT_[XYY2]AXIS_MAX	XtCXrtAxisBounds	XtRFloat
XtNxrt[XYY2]AxisMaxUseDefault XRT_[XYY2]AXIS_MAX_USE_DEFAULT	XtCXrtAxisBoundsUseDefault	XtRBoolean
XtNxrt[XYY2]AxisMin XRT_[XYY2]AXIS_MIN	XtCXrtAxisBounds	XtRFloat
XtNxrt[XYY2]AxisMinUseDefault XRT_[XYY2]AXIS_MIN_USE_DEFAULT	XtCXrtAxisBoundsUseDefault	XtRBoolean
XtNxrt[XYY2]AxisReversed XRT_[XYY2]AXIS_REVERSED	XtCXrtAxisReversed	XtRBoolean
XtNxrt[XYY2]AxisShow XRT_[XYY2]AXIS_SHOW	XtCXrtAxisShow	XtRBoolean

Name	Class	Type
XtNxrt[XYY2]Grid XRT_[XYY2]GRID	XtCXrtGrid	XtRFloat
XtNxrt[XYY2]GridDataStyle XRT_[XYY2]GRID_DATA_STYLE	XtCXrtDataStyle	XtRXrtDataStyle
XtNxrt[XYY2]GridDataStyleUseDefault XRT_[XYY2]GRID_DATA_STYLE_USE_DEFAULT	XtCXrtDataStylesUseDefault	XtRBoolean
XtNxrt[XYY2]GridUseDefault XRT_[XYY2]GRID_USE_DEFAULT	XtCXrtGridUseDefault	XtRBoolean
XtNxrt[XYY2]Labels XRT_[XYY2]LABELS	XtCXrtValueLabels	XtRXrtValueLabels
XtNxrt[XY]Marker XRT_[XY]MARKER	XtCXrtMarker	XtRFloat
XtNxrt[XY]MarkerDataStyle XRT_[XY]MARKER_DATA_STYLE	XtCXrtDataStyle	XtRXrtDataStyle
XtNxrt[XY]MarkerDataStyleUseDefault XRT_[XY]MARKER_DATA_STYLE_USE_DEFAULT	XtCXrtDataStylesUseDefault	XtRBoolean
XtNxrtXMarkerMethod XRT_XMARKER_METHOD	XtCXrtMarkerMethod	XtRXrtMarkerMethod
XtNxrtXMarkerPoint XRT_XMARKER_POINT	XtCXrtMarkerPoint	XtRInt
XtNxrtXMarkerSet XRT_XMARKER_SET	XtCXrtMarkerSet	XtRInt
XtNxrt[XY]MarkerShow XRT_[XY]MARKER_SHOW	XtCXrtMarkerShow	XtRBoolean
XtNxrt[XYY2]Max XRT_[XYY2]MAX	XtCXrtAxisBounds	XtRFloat
XtNxrt[XYY2]MaxUseDefault XRT_[XYY2]MAX_USE_DEFAULT	XtCXrtAxisBoundsUseDefault	XtRBoolean
XtNxrt[XYY2]Min XRT_[XYY2]MIN	XtCXrtAxisBounds	XtRFloat
XtNxrt[XYY2]MinUseDefault XRT_[XYY2]MIN_USE_DEFAULT	XtCXrtAxisBoundsUseDefault	XtRBoolean
XtNxrt[XYY2]Num XRT_[XYY2]NUM	XtCXrtAxisParams	XtRFloat
XtNxrt[XYY2]NumMethod XRT_[XYY2]NUM_METHOD	XtCXrtNumMethod	XtRXrtNumMethod
XtNxrt[XYY2]NumUseDefault XRT_[XYY2]NUM_USE_DEFAULT	XtCXrtAxisParamsUseDefault	XtRBoolean
XtNxrt[XYY2]Origin XRT_[XYY2]ORIGIN	XtCXrtAxisParams	XtRFloat

Name	Class	Type
XtNxrtXOriginBase XRT_XORIGIN_BASE	XtCXrtAxisParams	XtRFloat
XtNxrt[XYY2]OriginPlacement XRT_[XYY2]ORIGIN_PLACEMENT	XtCXrtOriginPlacement	XtRXrtOriginPlacement
XtNxrt[XYY2]OriginUseDefault XRT_[XYY2]ORIGIN_USE_DEFAULT	XtCXrtAxisParamsUseDefault	XtRBoolean
XtNxrt[XYY2]Precision XRT_[XYY2]PRECISION	XtCXrtPrecision	XtRInt
XtNxrt[XYY2]PrecisionUseDefault XRT_[XYY2]PRECISION_USE_DEFAULT	XtCXrtPrecisionUseDefault	XtRBoolean
XtNxrt[XYY2]Tick XRT_[XYY2]TICK	XtCXrtAxisParams	XtRFloat
XtNxrt[XYY2]TickUseDefault XRT_[XYY2]TICK_USE_DEFAULT	XtCXrtAxisParamsUseDefault	XtRBoolean
XtNxrt[XYY2]Title XRT_[XYY2]TITLE	XtCXrtTitle	XtRXrtString
XtNxrt[XYY2]TitleRotation XRT_[XYY2]TITLE_ROTATION	XtCXrtRotation	XtRXrtRotate
XtNxrt[YY2]AnnotationAngle XRT_[YY2]ANNOTATION_ANGLE	XtCXrtAxisParams	XtRFloat
XtNxrt[YY2]AnnotationAngleUseDefault XRT_[YY2]ANNOTATION_ANGLE_USE_DEFAULT	XtCXrtAxisParamsUseDefault	XtRBoolean
XtNxrt[YY2]Axis100Percent XRT_[YY2]AXIS_100_PERCENT	XtCXrtAxisParams	XtRBoolean
XtNxrtYAxisConst XRT_YAXIS_CONST	XtCXrtAxisConst	XtRFloat
XtNxrtYAxisMult XRT_YAXIS_MULT	XtCXrtAxisMult	XtRFloat
XtNxrtZoomAxisCallback[a]	XtCCallback	XtRCallback

a: このリソースは、Windows では使用されないのでクロス・プラットフォーム名がありません。

H

リソース･ファイルの使用

構文

リソース･ファイルの例

リソース･ファイル、GUI ビルダ、*XrtSetPropString()*に使用されるリソースの構文の一覧です。この章の最後には、サンプルのリソース･ファイルが提供されています。XRT/graph は、Xt と Motif に定義されていないデータ型のリソース変換のみを定義します。

`UseDefault` を除いて全ての XRT/graph のリソースは、リソース･ファイルで定義できます。

H.1 構文

Comments

行にあるエクスクラメーション･マーク(!)の後のテキストは無視されます。

Continuation(継続)

行の最後の文字をバック･スラッシュ(\)にすると、次に続く行に継続します。

Booleans

大文字、小文字、大小混合文字の“`True`”または“`False`”です。

列挙された型

列挙された型を用いるリソースは、列挙された値に基づく一貫した命名規則を使用します。全てのリソースでは、先頭の `XRT_`がなく、残りのアンダー･スコアもありません。リソース値は、大文字、小文字または大小混合の文字で入力できます。例えば、次の各リソースの指定は、どれも `XRT_BORDER_SHADOW` が `XtNxrtGraphBorder` で使用されるようにしています。

```
*.XtXrtGraph.xrtGrpahBorder: BorderShadow
*.XtXrtGraph.xrtGraphBorder: BORDERSHADOW
*.XtXrtGrpah.xrtGraphBorder: bordershadow
```

データ

XtRXrtData 型のリソースは、有効なファイル名で指定されます。リソース変換プログラムは、このファイルを開き、*XrtDataCreateFromFile()*を呼び出します。エラー・メッセージは、*stderr* に書かれます。

```
*.XtXrtGraph.xrtData: data/net_data.dat
```

データ・スタイル

一つのデータ・スタイル(XtRXrtDataStyle 型のリソース)は、括弧に囲まれた七つの値で指定されます。値は、線のパターン、塗りつぶしパターン、色、線幅、ポイント・スタイル、ポイントの色、ポイントのサイズです。次の例は、円グラフの“その他”のスライスのデフォルトのスタイルです。

```
*.bar.xrtOtherDataStyle: (LpatSolid FpatSolid \
  “yellow” 1 PointNone “black” 7)
```

データ・スタイルのリスト

XtRXrtDataStyles 型のリソースは、括弧に囲まれたリストとして指定され、各メンバが完全なデータ・スタイル(XtRXrtDataStyle)を指定しています。

```
*.bar.xrtDataStyles: ((LpatSolid FpatSolid \
    "#6CCC1D" 3 PointTri "Dark Orchid" 9 )\
  (LpatSolid FpatSolid "#5A8DCC" 3 PointBox \
    "Indian Red" 9 )\
  (LpatSolid FpatSolid "LimeGreen" 1 PointDot \
    "medium blue" 7 )\
  (LpatSolid FpatSolid "Medium Turquoise" 1 \
    PointDiamond "red" 7 )\
  (LpatSolid FpatSolid "blue" 1 PointStar \
    "red" 7 ))
*.bar.xrtDataStyles2: ()
```

イメージ

XtRXrtImage 型のリソースは、有効なファイル名で指定されます。ファイル名がチェックされ、デフォルトのイメージ型かユーザによって登録された型に変換されます。

```
*.XtXrtGraph.xrtLegendImage: $XRTHOME/src/icons/shading.xpm
```

ストリング値

指定されるストリング(XtRXrtString 型のリソース)がスペースを含む時は、必ず引用符(“)で囲んでください。リソース変換プログラムは、(“)を取り去ります。引用符をストリングの一部と指定する場合は、もう一つの引用符を前に付けてください。(“ “)

ストリングのリスト

XtRXrtStrings 型のリソースは、括弧で囲まれたストリング(XtRXrtString)のリストとして指定されます。空のリストを指定する場合は、一組の括弧を用いてください。

```
*.xrtSetLabels: (Eggs “Ham and Cheese” Rye)
*.xrtPointLabels: ()
```

値ラベル(Value-label)

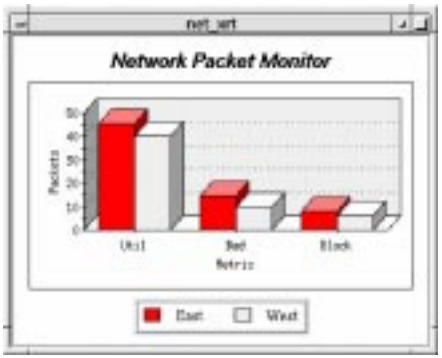
XtRXrtValueLabels 型のリソースは、括弧に囲まれたリストとして指定され、各メンバがそれぞれの括弧の中で完全な値ラベルを指定します。ラベルのストリングは、引用符に囲まれます。

```
*.XtXrtGraph.xrtXLabels: (\
    (2.3 “Cold”)\
    (3.5 “Warm”)\
    (5.2 “Hot”)\
)
```

変換プログラムは、値の増加に従って値ラベルをソートします。二重のエントリは、任意に削除されます。空のリストを指定するには、一組の括弧を用いてください。

H.2 リソース・ファイルの例

下記のリソース・ファイルは、次の図のグラフを定義しています。




```
*.net.xrtAxisBoundingBox: True
*.net.xrtAxisFont: fixed
*.net.xrtBackgroundColor: "White"
*.net.xrtBarClusterOverlap: 100
*.net.xrtBarClusterWidth: 70
*.net.xrtDataAreaBackgroundColor: "grey95"
*.net.xrtDataStyles: ((LpatSolid FpatSolid "red" 1 PointTri "magenta" 7 )\
(LpatSolid FpatSolid "grey95" 1 PointBox "green" 7 ))
*.net.xrtDataStyles2: ()
*.net.xrtDoubleBuffer: True
*.net.xrtFooterAdjust: AdjustCenter
*.net.xrtFooterBackgroundColor: ""
*.net.xrtFooterBorder: BorderNone
*.net.xrtFooterBorderWidth: 2
*.net.xrtFooterFont: fixed
*.net.xrtFooterForegroundColor: ""
*.net.xrtFooterStrings: ()
*.net.xrtFooterX: 198
*.net.xrtFooterY: 298
*.net.xrtForegroundColor: "Black"
*.net.xrtFrontDataset: 1
*.net.xrtGraphBackgroundColor: ""
*.net.xrtGraphBorder: BorderPlain
*.net.xrtGraphBorderWidth: 1
*.net.xrtGraphDepth: 10
*.net.xrtGraphForegroundColor: ""
*.net.xrtGraphHeight: 199
*.net.xrtGraphInclination: 30
*.net.xrtGraphMarginBottom: 55
*.net.xrtGraphMarginLeft: 51
*.net.xrtGraphMarginRight: 10
*.net.xrtGraphMarginTop: 15
*.net.xrtGraphRotation: 30
*.net.xrtGraphWidth: 364
*.net.xrtGraphX: 16
*.net.xrtGraphY: 44
```

```
*.net.xrtHeaderAdjust: AdjustCenter
*.net.xrtHeaderBackgroundColor: ""
*.net.xrtHeaderBorder: BorderNone
*.net.xrtHeaderBorderWidth: 2
*.net.xrtHeaderFont: -adobe-helvetica-bold-o-normal--18-180-75-75-p-104-iso8859-1
*.net.xrtHeaderForegroundColor: ""
*.net.xrtHeaderStrings: ("Network Packet Monitor")
*.net.xrtHeaderX: 94
*.net.xrtHeaderY: 15
*.net.height: 298
*.net.xrtInvertOrientation: False
*.net.xrtLegendAnchor: AnchorSouth
*.net.xrtLegendBackgroundColor: ""
*.net.xrtLegendBorder: BorderEtchedOut
*.net.xrtLegendBorderWidth: 5
*.net.xrtLegendFont: -adobe-times-medium-r-normal--14-140-75-75-p-74-iso8859-1
*.net.xrtLegendForegroundColor: ""
*.net.xrtLegendOrientation: AlignHorizontal
*.net.xrtLegendShow: True
*.net.xrtLegendX: 115
*.net.xrtLegendY: 251
*.net.xrtMarkerDataStyle: (LpatShortDash FpatSolid "black" 1 PointNone "black" 7 )
*.net.xrtOtherDataStyle: (LpatSolid FpatSolid "yellow" 1 PointNone "black" 7 )
*.net.xrtOtherLabel: "Other"
*.net.xrtPieMinSlices: 1
*.net.xrtPieOrder: PieOrderDataOrder
*.net.xrtPieThresholdMethod: PieSliceCutoff
*.net.xrtPieThresholdValue: 0.000000
*.net.xrtPointLabels: ("Util" "Bad" "Block")
*.net.xrtPointLabels2: ()
*.net.xrtSetLabels: ("East" "West")
*.net.xrtSetLabels2: ()
*.net.xrtTimeBase: 0
*.net.xrtTimeFormat: ""
*.net.xrtTimeUnit: TmunitSeconds
*.net.xrtType: TypeBar
*.net.xrtType2: TypeBar
*.net.width: 396
*.net.xrtXAnnoPlacement: AnnoAuto
*.net.xrtXAnnotationMethod: AnnoPointLabels
*.net.xrtXAnnotationRotation: RotateNone
*.net.xrtXAxisLogarithmic: False
*.net.xrtXAxisMax: 1.000000
*.net.xrtXAxisMin: 0.000000
*.net.xrtXAxisReversed: False
*.net.xrtXAxisShow: True
*.net.xrtXGrid: 0.000000
*.net.xrtXGridDataStyle: (LpatDotted FpatSolid "Black" 1 PointNone "black" 7 )
*.net.xrtXLabels: ()
*.net.xrtXMax: 0.000000
*.net.xrtXMin: 0.000000
*.net.xrtXNum: 1.000000
*.net.xrtXNumMethod: NumRound
*.net.xrtXOrigin: 0.000000
*.net.xrtXOriginPlacement: OriginAuto
*.net.xrtXPrecision: 0
*.net.xrtXTick: 0.500000
```

```
*.net.xrtXTitle: "Metric"
*.net.xrtXTitleRotation: RotateNone
*.net.xrtY2AnnotationMethod: AnnoValues
*.net.xrtY2AnnotationRotation: RotateNone
*.net.xrtY2AxisLogarithmic: False
*.net.xrtY2AxisMax: 340282346638528859811704183484516925440.000000
*.net.xrtY2AxisMin: 340282346638528859811704183484516925440.000000
*.net.xrtY2AxisShow: True
*.net.xrtY2Labels: ()
*.net.xrtY2Max: 340282346638528859811704183484516925440.000000
*.net.xrtY2Min: 340282346638528859811704183484516925440.000000
*.net.xrtY2Num: 340282346638528859811704183484516925440.000000
*.net.xrtY2NumMethod: NumRound
*.net.xrtY2Precision: 27241
*.net.xrtY2Tick: 340282346638528859811704183484516925440.000000
*.net.xrtY2Title: ""
*.net.xrtYAnnoPlacement: AnnoAuto
*.net.xrtYAnnotationMethod: AnnoValues
*.net.xrtYAnnotationRotation: RotateNone
*.net.xrtY2AxisReversed: False
*.net.xrtYAxisConst: 0.000000
*.net.xrtYAxisLogarithmic: False
*.net.xrtYAxisMax: 50.000000
*.net.xrtYAxisMin: 0.000000
*.net.xrtYAxisMult: 0.000000
*.net.xrtYAxisReversed: False
*.net.xrtYAxisShow: True
*.net.xrtYGrid: 10.000000
*.net.xrtYGridDataStyle: (LpatDotted FpatSolid "Black" 1 PointNone "black" 7 )
*.net.xrtYLabels: ()
*.net.xrtYMax: 45.599998
*.net.xrtYMin: 6.200000
*.net.xrtYNum: 10.000000
*.net.xrtYNumMethod: NumRound
*.net.xrtYOrigin: 0.000000
*.net.xrtYOriginPlacement: OriginAuto
*.net.xrtYPrecision: -1
*.net.xrtYTick: 5.000000
*.net.xrtYTitle: "Packets"
*.net.xrtY2TitleRotation: RotateNone
*.net.xrtYTitleRotation: Rotate90
```

I

UIL 引数

XRT/graph の UIL 引数とそのデータ型のアルファベット順のリストです。二番めの表は、XRT/graph の UIL の reason です。UIL に使用については、7.10 を参照してください。

UIL Argument Name	Argument Type
XtNxrtAngleUnit	integer
XtNxrtAxisBoundingBox	boolean
XtNxrtAxisFont	font_table
XtNxrtBackgroundColor	string
XtNxrtBarClusterOverlap	integer
XtNxrtBarClusterWidth	integer
XtNxrtBorder	integer
XtNxrtBubbleMax	float
XtNxrtBubbleMethod	XrtBubbleMethod
XtNxrtBubbleMin	XtNxrtBubbleMin
XtNxrtCandleComplex	boolean
XtNxrtDataAreaBackgroundColor	string
XtNxrtDataAreaForegroundColor	string
XtNxrtDataAreaImage	XrtImage
XtNxrtDataAreaImageLayout	Integer
XtNxrtDataStyles2UseDefault	Boolean
XtNxrtDataStylesUseDefault	Boolean
XtNxrtDebug	Boolean

UIL Argument Name	Argument Type
XtNxrtDoubleBuffer	boolean
XtNxrtExtraDefaultDataStyles	boolean
XtNxrtFooterAdjust	integer
XtNxrtFooterBackgroundColor	string
XtNxrtFooterBorder	integer
XtNxrtFooterBorderWidth	integer
XtNxrtFooterFont	font_table
XtNxrtFooterForegroundColor	string
XtNxrtFooterImage	XrtImage
XtNxrtFooterImageLayout	integer
XtNxrtFooterImageMinimumSize	boolean
XtNxrtFooterX	integer
XtNxrtFooterXUseDefault	boolean
XtNxrtFooterY	integer
XtNxrtFooterYUseDefault	boolean
XtNxrtForegroundColor	string
XtNxrtFrontDataset	integer
XtNxrtGraph3dShading	integer
XtNxrtGraphBackgroundColor	string
XtNxrtGraphBorder	integer
XtNxrtGraphBorderWidth	integer
XtNxrtGraphDepth	integer
XtNxrtGraphForegroundColor	string
XtNxrtGraphHeight	integer
XtNxrtGraphHeightUseDefault	boolean
XtNxrtGraphImage	XrtImage
XtNxrtGraphImageLayout	integer
XtNxrtGraphInclination	integer
XtNxrtGraphMarginBottom	integer
XtNxrtGraphMarginBottomUseDef[a]	boolean
XtNxrtGraphMarginLeft	integer
XtNxrtGraphMarginLeftUseDefault	boolean

UIL Argument Name	Argument Type
XtNxrtGraphMarginRight	integer
XtNxrtGraphMarginRightUseDef[a]	boolean
XtNxrtGraphMarginTop	integer
XtNxrtGraphMarginTopUseDefault	boolean
XtNxrtGraphRotation	integer
XtNxrtGraphShowOutlines	boolean
XtNxrtGraphWidth	integer
XtNxrtGraphWidthUseDefault	boolean
XtNxrtGraphX	integer
XtNxrtGraphXUseDefault	boolean
XtNxrtGraphY	integer
XtNxrtGraphYUseDefault	boolean
XtNxrtHeaderAdjust	integer
XtNxrtHeaderBackgroundColor	string
XtNxrtHeaderBorder	integer
XtNxrtHeaderBorderWidth	integer
XtNxrtHeaderFont	font_table
XtNxrtHeaderForegroundColor	string
XtNxrtHeaderImage	XrtImage
XtNxrtHeaderImageLayout	integer
XtNxrtHeaderImageMinimumSize	boolean
XtNxrtHeaderX	integer
XtNxrtHeaderXUseDefault	boolean
XtNxrtHeaderY	integer
XtNxrtHeaderYUseDefault	boolean
XtNxrtHiLoCloseShow	boolean
XtNxrtHiLoOpenCloseFullWidth	boolean
XtNxrtHiLoOpenShow	boolean
XtNxrtImage	XrtImage
XtNxrtImageLayout	Integer
XtNxrtInvertOrientation	boolean
XtNxrtIsStacked	boolean

UIL Argument Name	Argument Type
XtNxrtIsStacked2	boolean
XtNxrtLegendAnchor	integer
XtNxrtLegendBackgroundColor	string
XtNxrtLegendBorder	integer
XtNxrtLegendBorderWidth	integer
XtNxrtLegendFont	font_table
XtNxrtLegendForegroundColor	string
XtNxrtLegendImage	XrtImage
XtNxrtLegendImageLayout	integer
XtNxrtLegendOrientation	integer
XtNxrtLegendReversed	boolean
XtNxrtLegendShow	boolean
XtNxrtLegendTitle	string
XtNxrtLegendX	integer
XtNxrtLegendXUseDefault	boolean
XtNxrtLegendY	integer
XtNxrtLegendYUseDefault	boolean
XtNxrtMarkerDataStyleUseDefault	boolean
XtNxrtMarkerDataset	integer
XtNxrtOtherDataStyleUseDefault	boolean
XtNxrtOtherLabel	string
XtNxrtPieMergeMissingSlices	boolean
XtNxrtPieMinSlices	integer
XtNxrtPieOrder	integer
XtNxrtPieStartAngle	float
XtNxrtPieThresholdMethod	integer
XtNxrtPieThresholdValue	single_float
XtNxrtPolarAxisAllowNegatives	boolean
XtNxrtPolarHalfRange	boolean
XtNxrtRepaint	boolean
XtNxrtTimeBase	integer
XtNxrtTimeFormat	string

UIL Argument Name	Argument Type
XtNxrtTimeFormatUseDefault	boolean
XtNxrtTimeUnit	integer
XtNxrtType	integer
XtNxrtType2	integer
XtNxrt[XYY2]AnnoPlacement	integer
XtNxrt[XYY2]AnnotationMethod	integer
XtNxrt[XYY2]AnnotationRotation	integer
XtNxrt[XYY2]AxisDataStyle	XrtDataStyle
XtNxrt[XYY2]AxisDataStyleUseDefault	boolean
XtNxrt[XYY2]AxisLabelFormat	string
XtNxrt[XYY2]AxisLogarithmic	boolean
XtNxrt[XYY2]AxisMax	single_float
XtNxrt[XYY2]AxisMaxUseDefault	boolean
XtNxrt[XYY2]AxisMin	single_float
XtNxrt[XYY2]AxisMinUseDefault	boolean
XtNxrt[XYY2]AxisReversed	boolean
XtNxrt[YY2]XaxisShow	boolean
XtNxrt[XYY2]Grid	single_float
XtNxrt[XYY2]GridDataStyleUseDefault	boolean
XtNxrt[XYY2]GridUseDefault	boolean
XtNxrt[XY]Marker	single_float
XtNxrt[XY]MarkerDataStyleUseDef[a]	boolean
XtNxrtXMarkerMethod	integer
XtNxrtXMarkerPoint	integer
XtNxrtXMarkerSet	integer
XtNxrt[XY]MarkerShow	boolean
XtNxrt[XYY2]Max	single_float
XtNxrt[XYY2]MaxUseDefault	boolean
XtNxrt[XYY2]Min	single_float
XtNxrt[XYY2]MinUseDefault	boolean
XtNxrt[XYY2]Num	single_float
XtNxrt[XYY2]NumMethod	integer

UIL Argument Name	Argument Type
XtNxrt[XYY2]NumUseDefault	boolean
XtNxrt[XYY2]Origin	single_float
XtNxrtXOriginBase	single_float
XtNxrt[XYY2]OriginPlacement	integer
XtNxrt[XYY2]OriginUseDefault	boolean
XtNxrt[XYY2]Precision	integer
XtNxrt[XYY2]PrecisionUseDefault	boolean
XtNxrt[XYY2]Tick	single_float
XtNxrt[XYY2]TickUseDefault	boolean
XtNxrt[XYY2]Title	string
XtNxrt[XYY2]TitleRotation	integer
XtNxrt[YY2]AnnotationAngle	single_float
XtNxrt[YY2]AnnotationAngleUseDef[a]	boolean
XtNxrt[YY2]Axis100Percent	boolean
XtNxrtYAxisConst	single_float
XtNxrtYAxisMult	single_float

a: 引数名は、UIL の 31 文字の制限に合わせるため、短くなっています。

Reasons
XtNxrtExposeCallback
XtNxrtMapCallback
XtNxrtModifyCallback
XtNxrtPickCallback
XtNxrtResizeCallback
XtNxrtRotateCallback
XtNxrtTransformCallback
XtNxrtZoomAxisCallback

索引

_

_xrt.c, 39
_xrt.h, 40
_xrt.make, 40
_xrt.res, 40
_xrt_main.c, 40

3

3D 表示, 70

A

AxisDataStyle, 79

C

C++, 11, 140

H

HiLoOpenClose グラフ, 84, 104

M

Motif との統合, 21
MS Windows へのポーティング, 136
MtGraphTransposeData(), 105

P

PointStyle, 209
Property Editor, 33

U

UIL, 141, 283
UseDefault, 27

X

XRT/graph 3.0
  用語の解説, 21
XRT/graph 4.0
  新しい機能, 2
xrt_auth, 10, 41
xrt_gm, 38
XRT_POINTSTYLE_DATA_STYLE, 210
XRT_POINTSTYLE_DATA_STYLE_USE_DEFAULT, 210
XRT_POINTSTYLE_DATASET, 209
XRT_POINTSTYLE_DISPLAY, 209
XRT_POINTSTYLE_PATTERN_BACKGROUND_COLOR, 210
XRT_POINTSTYLE_POINT, 209
XRT_POINTSTYLE_SET, 209
XRT_POINTSTYLE_SLICE_OFFSET, 209
XRT_POINTSTYLE_SLICE_OFFSET_USE_DEFAULT, 210
XRT_POINTSTYLE_USE_DEFAULT, 210
XRT_TEXT_ADJUST, 205
XRT_TEXT_ANCHOR, 205
XRT_TEXT_ATTACH_DATASET, 205
XRT_TEXT_ATTACH_PIXEL_X, 205
XRT_TEXT_ATTACH_PIXEL_Y, 205
XRT_TEXT_ATTACH_POINT, 206
XRT_TEXT_ATTACH_SET, 206
XRT_TEXT_ATTACH_TYPE, 206
XRT_TEXT_ATTACH_VALUE_X, 206
XRT_TEXT_ATTACH_VALUE_Y, 206
XRT_TEXT_BACKGROUND_COLOR, 206
XRT_TEXT_BORDER, 206
XRT_TEXT_BORDER_WIDTH, 206
XRT_TEXT_FONT, 206
XRT_TEXT_FOREGROUND_COLOR, 206
XRT_TEXT_HEIGHT, 208
XRT_TEXT_IMAGE, 207
XRT_TEXT_IMAGE_LAYOUT, 207
XRT_TEXT_IMAGE_MINIMUM_SIZE, 207
XRT_TEXT_IS_CONNECTED, 207
XRT_TEXT_IS_SHOWING, 207
XRT_TEXT_NAME, 207
XRT_TEXT_OFFSET, 207

XRT_TEXT_PRINT_FONT, 207
XRT_TEXT_ROTATION, 207
XRT_TEXT_STRINGS, 208
XRT_TEXT_WIDTH, 208
XRT_TEXT_X, 208
XRT_TEXT_Y, 208
XrtAdjust, 257
XrtAlign, 257
XrtAnchor, 257
XrtAngleUnit(), 257
XrtAnnoMethod, 258
XrtAnnoPlacement, 258
XrtArgValToFloat(), 211
XrtArrCheckAxisBounds(), 211
XrtArrDataAppendPts(), 212
XrtArrDataFastUpdate(), 212
XrtArrDataRemovePts(), 213
XrtArrDataShiftPts(), 213
XrtAttachType, 258
XrtAxis, 258
XrtBorder, 258
XrtCallbackStrunct, 259
XrtColor, 259
XrtCustomFormatValidate(), 69, 213
XrtDataConcat(), 213
XrtDataConvertToHandle(), 214
XrtDataCopy(), 214
XrtDataCreate(), 97, 214
XrtDataCreateFromFile(), 13, 97, 215
XrtDataDestroy(), 216
XrtDataExtractSet(), 217
XrtDataGetDisplay(), 104, 217
XrtDataGetFirstPoint(), 217
XrtDataGetFirstSet(), 217
XrtDataGetHole(), 217
XrtDataGetLastPoint(), 218
XrtDataGetLastSet(), 218
XrtDataGetNPoints(), 218
XrtDataGetNSets(), 218
XrtDataGetType(), 218
XrtDataGetXData(), 102, 219
XrtDataGetXElement(), 219
XrtDataGetYData(), 102, 219
XrtDataGetYElement(), 219
XrtDataHandle, 98, 100
XrtDataRelease(), 103, 220
XrtDataRemoveSet(), 220
XrtDataSaveToFile(), 220
XrtDataSetDisplay(), 104, 220
XrtDataSetFirstPoint(), 221
XrtDataSetFirstSet(), 221
XrtDataSetHole(), 221
XrtDataSetLastPoint(), 221
XrtDataSetLastSet(), 222
XrtDataSetNPoints(), 222
XrtDataSetNSets(), 222
XrtDataSetType(), 222
XrtDataSetXData(), 223
XrtDataSetXElement(), 223
XrtDataSetYData(), 223
XrtDataSetYElement(), 224
XrtDataSort(), 224
XrtDataStyle, 259
XrtDataTranspose(), 104, 224
XrtDataUpdateDataValue(), 224
XrtDeletePointLabel(), 225
XrtDeletePointLabel2(), 225
XrtDeleteSetLabel(), 225
XrtDeleteSetLabel2(), 225
XrtDestroyData(), 225
XrtDisplay, 259
XrtDrawPS(), 95, 225
XrtDrawPSColor, 259
XrtDupDataStyles(), 228
XrtDupStrings(), 228
XrtDupValueLabels(), 228
XrtEnableImageType(), 228
XrtFillPattern, 260
XrtFloatToArgVal(), 229
XrtFreeDataStyles(), 229
XrtFreePropString(), 229
XrtFreeStrings(), 229
XrtFreeTextHandles(), 229
XrtFreeValueLabels(), 229
XrtGenCheckAxisbounds(), 230
XrtGenDataAppendPt(), 230
XrtGenDataFastUpdate(), 231
XrtGenDataRemovePt(), 231
XrtGenDataShiftPts(), 231
XrtGenerateCode(), 39
XrtGetImage(), 95, 232
XrtGetNthDataStyle(), 78, 232
XrtGetNthDataStyle2(), 232
XrtGetNthFooterString(), 232
XrtGetNthHeaderString(), 233
XrtGetNthPointLabel(), 233
XrtGetNthPointLabel2(), 233
XrtGetNthSetLabel(), 233
XrtGetNthSetLabel2(), 233
XrtGetOtherSliceSets(), 82, 234
XrtGetPropString(), 234
XrtGetTextHandles(), 234
XrtGetValueLabel(), 234
XrtGetValues(), 235

XRTHOME, 9
XrtImage, 260
XrtInsertNthDataStyle(), 235
XrtInsertNthPointLabel(), 235
XrtInsertNthPointLabel2(), 235
XrtInsertNthSetLabel(), 236
XrtInsertNthSetLabel2(), 236
XrtLinePattern, 260
XrtLoadGraphFromFile(), 37
XrtMakeData(), 236
XrtMakeDataFromFile(), 236
XrtMakeTime(), 237
XrtMap(), 118, 123, 237
XrtMapCallbackStruct, 260
XrtMapResult, 260
XrtMarkerMethod, 261
XrtModifyCallbackStruct, 261
XrtMrmInitialize(), 237
XrtNumMethod, 261
XrtOriginPlacement, 261
XrtOutputPS(), 95, 238
XrtOutputRaster(), 95, 238
XrtOutputXwd(), 95, 238
XrtPDSCmpDoubles(), 158
XrtPDSCmpFloats(), 159
XrtPDSCmpInts(), 159
XrtPDSCmpStrings(), 159
XrtPDSCmpStringsIgnoreCase(), 159
XrtPDSImage, 143
XrtPDSImageCopy(), 156
XrtPDSImageCreate(), 156
XrtPDSImageCreateProc, 147
XrtPDSImageDestroy(), 156
XrtPDSImageFuncs, 144
XrtPDSImageGetClipMask(), 157
XrtPDSImageGetColorDepth(), 157
XrtPDSImageGetFilename(), 157
XrtPDSImageGetHeight(), 157
XrtPDSImageGetImage(), 158
XrtPDSImageGetWidth(), 157
XrtPDSImageHeaderCompareProc, 147
XrtPDSImageIsClipMaskSet(), 158
XrtPDSImageIsImage(), 158
XrtPDSImageIsSet(), 158
XrtPDSImageNameCompareProc, 147
XrtPDSImageOutputProc, 147
XrtPDSList, 148
XRTPDSListAdd(), 160
XrtPDSListAppend(), 160
XrtPDSListCreate(), 160
XrtPDSListDelete(), 160
XrtPDSListDeleteALL(), 161
XrtPDSListDestroy(), 161
XrtPDSListDestroyObject(), 161
XrtPDSListDestroyXtPoiter(), 161
XrtPDSListDuplicate(), 161
XrtPDSListFind(), 162
XrtPDSListGetCompareProc(), 162
XrtPDSListGetItem(), 162
XrtPDSListGetItemCount(), 162
XrtPDSListGetItems(), 162
XrtPDSListGetItemSize(), 163
XrtPDSListGetUserData(), 163
XrtPDSListInsertAfter(), 163
XrtPDSListInsertBefore(), 163
XrtPDSListIsList(), 164
XrtPDSListIsSorted(), 164
XrtPDSListLookupItem(), 164
XrtPDSListMoveItemsAfter(), 164
XrtPDSListMoveItemsBefore(), 165
XrtPDSListRemove(), 165
XrtPDSListRemoveAll(), 165
XrtPDSListSearchForward(), 165
XrtPDSListSearchReverse(), 166
XrtPDSListSetCompareProc(), 166
XrtPDSListSetDestroyProc(), 167
XrtPDSListSetItem(), 167
XrtPDSListSetUserData(), 167
XrtPDSListSort(), 167
XrtPick(), 117, 120, 239
XrtPickCallbackStruct, 261
XrtPickResult, 261
XrtPieOrder, 262
XrtPieThresholdMethod, 262
XrtPoint, 262
XrtPointStyleCreate(), 239
XrtPointStyleDestroy(), 239
XrtPointStyleGetValues(), 240
XrtPointStyleHandle, 79
XrtPointStyleHandle(), 240
XrtPointStyleSetValues(), 240
XrtPopdownPropertyEditor(), 37
XrtPopupPropertyEditor(), 37
XrtProperty, 262
XrtPropertyEditorEnable(), 36, 241
XrtReason, 262
XrtRegion, 263
XrtRemoveNthDataStyle(), 241
XrtRotate, 263
XrtRotateCallbackStruct, 263
XrtSamleColorFunc(), 241
XrtSaveDataToFile(), 241
XrtSaveGraphToFile(), 37
XrtSaveImageAsJpeg(), 242

XrtSaveImageAsPNG(), 242
XrtSaveImageGeneric(), 243
XrtSetNthDataStyle(), 78, 243
XrtSetNthDataStyle2(), 243
XrtSetNthFooterString(), 243
XrtSetNthHeaderString(), 244
XrtSetNthPointLabel(), 244
XrtSetNthPointLabel2(), 244
XrtSetNthSetLabel(), 244
XrtSetNthSetLabel2(), 244
XrtSetPropString(), 244, 277
XrtSetReverseConverters(), 141, 245
XrtSetValueLabel(), 245
XrtSetValues(), 245
XrtShading, 263
XrtTextAreaCreate(), 245
XrtTextAreaDestroy(), 246
XrtTextAreaGetPropString(), 246
XrtTextAreaGetValues(), 246
XrtTextAreaSetPropString(), 246
XrtTextAreaSetValues(), 131, 247
XrtTextAttach(), 247
XrtTextCreate(), 247
XrtTextDestroy(), 247
XrtTextDetach(), 247
XrtTextDetail(), 248
XrtTextHandle, 130
XrtTextUpdate(), 248
XrtTimeUnit, 263
XrtTransformCallbackStruct, 264
XrtType, 264
XrtUnmap(), 124, 248
XrtUnpick(), 122, 248
XrtUpdatePropertyEditor(), 37
XrtValueLabel, 54, 264
XrtZoomAxisCallbackStruct, 265
XtAddTimeOut(), 98
XtAppAddInput(), 98
XtCreateXrtGraph(), 249
XtIsXrtGraph(), 249
XtNxrt[XY]MarkerShow, 90
XtNxrt[XY]Origin, 61
XtNxrt[XY]OriginPlacement, 61
XtNxrt[XYY2]AnnoPlacement, 65
XtNxrt[XYY2]AnnotationMethod, 50
XtNxrt[XYY2]AnnotationRotation, 63
XtNxrt[XYY2]AxisDataStyle, 66, 79
XtNxrt[XYY2]AxisLabelFormat, 52, 66
XtNxrt[XYY2]AxisLogarithmic, 52
XtNxrt[XYY2]AxisMax, 60, 113
XtNxrt[XYY2]AxisMin, 60, 113
XtNxrt[XYY2]AxisReversed, 64
XtNxrt[XYY2]AxisShow, 65
XtNxrt[XYY2]Grid, 62
XtNxrt[XYY2]GridDataStyle, 62
XtNxrt[XYY2]Max, 60
XtNxrt[XYY2]Min, 60
XtNxrt[XYY2]NumMethod, 50
XtNxrt[XYY2]Precision, 51
XtNxrt[XYY2]Tick, 62
XtNxrt[XYY2]Title, 63
XtNxrt[XYY2]TitleRotation, 63
XtNxrt[YY2]AnnotationAngle, 88
XtNxrt[YY2]Axis100Percent, 72, 81
XtNxrtAngleUnit, 87, 171
XtNxrtAxisBoundingBox, 66, 172
XtNxrtAxisFont, 172
XtNxrtBackgroundColor, 88, 172
XtNxrtBarClusterOverlap, 81, 172
XtNxrtBarClusterWidth, 80, 172
XtNxrtBorder, 172
XtNxrtBubbleMax, 85, 172
XtNxrtBubbleMethod, 84, 172
XtNxrtBubbleMin, 85, 172
XtNxrtCandleComplex, 86, 173
XtNxrtData, 98, 105, 173
XtNxrtData2, 173
XtNxrtDataAreaBackgroundColor, 88, 173
XtNxrtDataAreaForegroundColor, 89, 173
XtNxrtDataAreaImage, 173
XtNxrtDataAreaImageLayout, 173
XtNxrtDataStyles, 174
XtNxrtDataStyles2, 174
XtNxrtDataStyles2UseDefault, 174
XtNxrtDataStylesUseDefault, 74, 174
XtNxrtDebug, 174
XtNxrtDoubleBuffer, 90, 93, 174
XtNxrtExposeCallback, 125, 174
XtNxrtExtraDefaultDataStyles, 174
XtNxrtFooterAdjust, 72, 175
XtNxrtFooterBackgroundColor, 175
XtNxrtFooterBorder, 175
XtNxrtFooterBorderWidth, 176
XtNxrtFooterFont, 176
XtNxrtFooterForegroundColor, 176
XtNxrtFooterHeight, 176
XtNxrtFooterImage, 176
XtNxrtFooterImageLayout, 176
XtNxrtFooterImageMinimumSize, 93, 176
XtNxrtFooterStrings, 72, 176
XtNxrtFooterWidth, 176
XtNxrtFooterX, 176
XtNxrtFooterXUseDefault, 176
XtNxrtFooterY, 176

XtNxrtFooterYUseDefault, 176
XtNxrtForegroundColor, 88, 177
XtNxrtFrontDataset, 129, 177
XtNxrtGraph3dShading, 71, 177
XtNxrtGraphBackgroundColor, 177
XtNxrtGraphBorder, 91, 177
XtNxrtGraphBorderWidth, 91, 177
XtNxrtGraphDepth, 70, 177
XtNxrtGraphForegroundColor, 178
XtNxrtGraphHeight, 179
XtNxrtGraphHeightUseDefault, 179
XtNxrtGraphImage, 178
XtNxrtGraphImageLayout, 178
XtNxrtGraphInclination, 70, 115, 178
XtNxrtGraphMarginBottom, 64, 178
XtNxrtGraphMarginBottomUseDefault, 178
XtNxrtGraphMarginLeft, 64, 178
XtNxrtGraphMarginLeftUseDefault, 178
XtNxrtGraphMarginRight, 64, 179
XtNxrtGraphMarginRightUseDefault, 179
XtNxrtGraphMarginTop, 64, 179
XtNxrtGraphMarginTopUseDefault, 179
XtNxrtGraphRotation, 70, 115, 179
XtNxrtGraphShowOutlines, 179
XtNxrtGraphWidth, 179
XtNxrtGraphWidthUseDefault, 179
XtNxrtGraphX, 180
XtNxrtGraphXUseDefault, 180
XtNxrtGraphY, 180
XtNxrtGraphYUseDefault, 180
XtNxrtHeaderAdjust, 72, 180
XtNxrtHeaderBackgroundColor, 180
XtNxrtHeaderBorder, 180
XtNxrtHeaderBorderWidth, 180
XtNxrtHeaderFont, 180
XtNxrtHeaderForegroundColor, 180
XtNxrtHeaderHeight, 181
XtNxrtHeaderImage, 181
XtNxrtHeaderImageLayout, 181
XtNxrtHeaderImageMinimumSize, 93, 181
XtNxrtHeaderStrings, 72, 181
XtNxrtHeaderWidth, 181
XtNxrtHeaderX, 181
XtNxrtHeaderXUseDefault, 181
XtNxrtHeaderY, 181
XtNxrtHeaderYUseDefualt, 181
XtNxrtHiLoCloseShow, 181
XtNxrtHiLoOpenCloseFullWidth, 181
XtNxrtHiLoOpenShow, 181
XtNxrtImage, 182
XtNxrtImageLayout, 182
XtNxrtInvertOrientation, 16, 63, 182
XtNxrtIsStacked, 72, 182
XtNxrtIsStacked2, 182
XtNxrtLegendAnchor, 69, 182
XtNxrtLegendBackgroundColor, 88, 182
XtNxrtLegendBorder, 182
XtNxrtLegendBorderWidth, 182
XtNxrtLegendFont, 182
XtNxrtLegendForegroundColor, 183
XtNxrtLegendHeight, 183
XtNxrtLegendImage, 183
XtNxrtLegendImageLayout, 183
XtNxrtLegendOrientation, 73, 183
XtNxrtLegendReversed, 183
XtNxrtLegendShow, 183
XtNxrtLegendTitle, 73, 183
XtNxrtLegendWidth, 183
XtNxrtLegendX, 183
XtNxrtLegendXUseDefault, 183
XtNxrtLegendY, 183
XtNxrtLegendYUseDefault, 183
XtNxrtMapCallback, 184
XtNxrtMarkerDataset, 89, 129, 184
XtNxrtMarkerDataStyle, 90, 184
XtNxrtMarkerDataStyleUseDefault, 184
XtNxrtModifyCallback, 112, 116, 184
XtNxrtOtherDataStyle, 82, 185
XtNxrtOtherDataStyleUseDefault, 185
XtNxrtOtherLabel, 82, 185
XtNxrtPickCallback, 185
XtNxrtPieMinSlices, 83, 186
XtNxrtPieOrder, 83, 186
XtNxrtPieStartAngle, 186
XtNxrtPieThresholdMethod, 82, 186
XtNxrtPieThresholdValue, 82, 186
XtNxrtPointLabels, 54, 187
XtNxrtPointLabels2, 187
XtNxrtPolarAxisAllowNegatives, 88, 187
XtNxrtPolarHalfRange, 88, 187
XtNxrtRepaint, 133, 187
XtNxrtResizeCallback, 125, 187
XtNxrtRotateCallback, 115, 187
XtNxrtSetLabels, 54, 188
XtNxrtSetLabels2, 188
XtNxrtTextImageMinimumSize, 93
XtNxrtTimeBase, 56, 188
XtNxrtTimeFormat, 57, 188
XtNxrtTimeFormatUseDefault, 189
XtNxrtTimeUnit, 56, 189
XtNxrtTransformCallback, 113, 190
XtNxrtType, 24, 190
XtNxrtType2, 190
XtNxrtXAnnoPlacement, 190

XtNxrtXAnnotationMethod, 191
XtNxrtXAnnotationRotation, 191
XtNxrtXAxisDataStyle, 191
XtNxrtXAxisDataStyleUseDefault, 191
XtNxrtXAxisLabelFormat, 192
XtNxrtXAxisLogarithmic, 192
XtNxrtXAxisMax, 193
XtNxrtXAxisMaxUseDefault, 193
XtNxrtXAxisMin, 193
XtNxrtXAxisMinUseDefault, 194
XtNxrtXAxisReversed, 194
XtNxrtXAxisShow, 194
XtNxrtXGrid, 194
XtNxrtXGridDataStyle, 195
XtNxrtXGridDataStyleUseDefault, 195
XtNxrtXGridUseDefault, 195
XtNxrtXLabels, 195
XtNxrtXMarker, 195
XtNxrtXMarkerDataStyle, 196
XtNxrtXMarkerDataStyleUseDefault, 196
XtNxrtXMarkerMethod, 90, 196
XtNxrtXMarkerPoint, 90, 196
XtNxrtXMarkerSet, 90, 196
XtNxrtXMarkerShow, 196
XtNxrtXMax, 197
XtNxrtXMaxUseDefault, 197
XtNxrtXMin, 197
XtNxrtXMinUseDefault, 197
XtNxrtXNum, 198
XtNxrtXNumMethod, 198
XtNxrtXNumUseDefault, 198
XtNxrtXOrigin, 198
XtNxrtXOriginBase, 87, 198
XtNxrtXOriginPlacement, 199
XtNxrtXOriginUseDefault, 199
XtNxrtXPrecision, 199
XtNxrtXPrecisionUseDefault, 199
XtNxrtXTick, 200
XtNxrtXTickUseDefault, 57, 200
XtNxrtXTitle, 200
XtNxrtXTitleRotation, 200
XtNxrtY2AnnoPlacement, 190
XtNxrtY2AnnotationAngle, 201
XtNxrtY2AnnotationAngleUseDefault, 201
XtNxrtY2AnnotationMethod, 191
XtNxrtY2AnnotationRotation, 191
XtNxrtY2Axis100Percent, 201
XtNxrtY2AxisDataStyle, 191
XtNxrtY2AxisDataStyleUseDefault, 191
XtNxrtY2AxisLabelFormat, 192
XtNxrtY2AxisLogarithmic, 128, 192
XtNxrtY2AxisMax, 193
XtNxrtY2AxisMaxUseDefault, 193
XtNxrtY2AxisMin, 193
XtNxrtY2AxisMinUseDefault, 194
XtNxrtY2AxisReversed, 194
XtNxrtY2AxisShow, 194
XtNxrtY2Grid, 194
XtNxrtY2GridDataStyle, 195
XtNxrtY2GridDataStyleUseDefault, 195
XtNxrtY2GridUseDefault, 195
XtNxrtY2Labels, 195
XtNxrtY2Max, 197
XtNxrtY2MaxUseDefault, 197
XtNxrtY2Min, 197
XtNxrtY2MinUseDefault, 197
XtNxrtY2Num, 198
XtNxrtY2NumMethod, 198
XtNxrtY2NumUseDefault, 198
XtNxrtY2Origin, 198
XtNxrtY2OriginPlacement, 199
XtNxrtY2OriginUseDefault, 199
XtNxrtY2Precision, 199
XtNxrtY2PrecisionUseDefault, 199
XtNxrtY2Tick, 200
XtNxrtY2TickUseDefault, 200
XtNxrtY2Title, 200
XtNxrtY2TitleRotation, 200
XtNxrtYAnnoPlacement, 190
XtNxrtYAnnotationAngle, 201
XtNxrtYAnnotationAngleUseDefault, 201
XtNxrtYAnnotationMethod, 191
XtNxrtYAnnotationRotation, 191
XtNxrtYAxis100Percent, 201
XtNxrtYAxisConst, 127, 201
XtNxrtYAxisDataStyle, 191
XtNxrtYAxisDataStyleUseDefault, 191
XtNxrtYAxisLabelFormat, 192
XtNxrtYAxisLogarithmic, 192
XtNxrtYAxisMax, 193
XtNxrtYAxisMaxUseDefault, 193
XtNxrtYAxisMin, 193
XtNxrtYAxisMinUseDefault, 194
XtNxrtYAxisMult, 127, 201
XtNxrtYAxisReversed, 194
XtNxrtYAxisShow, 194
XtNxrtYGrid, 194
XtNxrtYGridDataStyle, 195
XtNxrtYGridDataStyleUseDefault, 195
XtNxrtYGridUseDefault, 195
XtNxrtYLabels, 195
XtNxrtYMarker, 195
XtNxrtYMarkerDataStyle, 196
XtNxrtYMarkerDataStyleUseDefault, 196

XtNxrtYMarkerShow, 196
XtNxrtYMax, 197
XtNxrtYMaxUseDefault, 197
XtNxrtYMin, 197
XtNxrtYMinUseDefault, 197
XtNxrtYNum, 198
XtNxrtYNumMethod, 198
XtNxrtYNumUseDefault, 198
XtNxrtYOrigin, 198
XtNxrtYOriginPlacement, 199
XtNxrtYOriginUseDefault, 199
XtNxrtYPrecision, 199
XtNxrtYPrecisionUseDefault, 199
XtNxrtYTick, 200
XtNxrtYTickUseDefault, 200
XtNxrtYTitle, 200
XtNxrtYTitleRotation, 200
XtNxrtZoomAxisCallback, 114, 201

Z

zoom rubberband, 112

あ

アクション・ルーチン
  Map(), 118, 252
  ModifyEnd(), 116, 253
  ModifyStart(), 111, 252
  NoAction(), 116, 253
  Pick(), 117, 253
  Properties(), 254
  RotateUpdate(), 115, 254
  ScaleUpdate(), 112, 254
  TranslateUpdate(), 112, 254
  ViewReset(), 114, 254
  ZoomAxisEnd(), 113, 255
  ZoomAxisUpdate(), 255
  ZoomCancel(), 255
  ZoomEnd(), 112, 255
  ZoomUpdate(), 112, 255
アプリケーションの許可, 10
イメージ
  型の登録, 144
  背景, 91, 133
色, 29, 88
  カラーマップのフラッシュ, 29
  近い色の取得, 30
印刷, 95, 226
インバート, 16
円グラフ, 82
  しきい値, 82
  スライスの最小数, 83
  スライスの順序, 83

か

環境変数, 9
キャンドル・グラフ, 84, 104
共用ライブラリ, 148
クラスタ, 80
グラフのタイプ, 24
グリッド線, 62
警告メッセージ, 140
原点, 61, 87
高速アップデート, 135
国際化サポート, 133

さ

サンプル・コード, 18
しきい値, 82
軸
  ズーム, 113
  精度, 128
  対数, 52
  タイトル, 63
  データスタイル, 66, 79
  二番目, 127
  反転, 64
  表示, 65
  方向, 63
  目盛り(tick), 62
  ラベルのフォント, 64
  最小と最大, 60
  時間, 56
軸の注釈, 50
  Value, 50, 66
  値ラベル, 54
  位置, 65
  インクレメント, 51
  回転, 63
  精度, 51, 58
  フォーマット, 52, 66
  ポイント・ラベル, 52
スクロールバー, 94

スタッキング(積上げ), 72
ソースコードの生成, 38

た

ダイナミック・ライブラリ, 148
ダブル・バッファリング, 93
積上げ棒グラフ, 80, 81
テキスト・オブジェクト, 130
  色, 133
  回転, 133
  削除, 133
  作成, 130
  属性, 130, 205
  フォント, 132
  枠, 133
  接続線, 132
  貼り付け, 131
データ, 257
  一般(GENERAL), 13, 99, 215
  除外と非表示, 103
  スタイル, 74
  静的, 106
  配列(ARRAY), 13, 99, 215
  範囲, 60
  フォーマット, 13, 99
  リアルタイム, 97, 107
  ロード, 11, 13, 36, 97
  最小と最大, 60
データ・オブジェクト, 98
データ・オブジェクト
  手続きと方法, 100
  複数, 105
デモ, 18
トランスポーズ, 104
トランスレーション・テーブル, 111, 251

は

背景
  イメージ, 91, 133
パフォーマンス, 63, 90, 133
バブル・チャート, 84
凡例, 73, 129
フォント, 73, 133
複合グラフ, 129
プロパティ・エディタ, 33, 109
  builder デモ, 34
  アプリケーションで使用, 34
  カスタマイズ, 36
  ソースコードの生成, 38
  プロパティの編集, 35
ヘッダーとフッター, 72
ポイントスタイル, 79, 209
棒グラフ, 80
  クラスタ, 80
  棒の重なり, 81
ポーラー・グラフ, 87

ま

マーカ, 89
マージン, 64

や

ユーザ・インタラクション, 109
  アクション・ルーチン, 252
  アップデート, 112
  開始, 111
  回転, 115
  グラフ・データ, 117
  コールバック, 111
  軸のズーム, 113
  終了, 116
  トランスレーション・テーブル, 111, 251
  不許可, 116
  リサイズ, 125
  リセット, 114

ら

リソース
  UseDefault, 27
  一覧, 171
  型とクラス, 267
  ストリング, 28
  バッチ処理, 133
  浮動小数点, 27
  変換, 141
  ポインタ, 28
リソースファイル, 31, 277

レーダー・グラフ, 87

わ

枠, 91