```
PROCEDURE ChannelMarker;
    CONST
        delimiter = ';';
        brightness = 0.75;
        minLeftColWidth = 3;
        minRightColWidth = 3;
        interColWidth = 8;
        maxColCount = 15;

        nothingSelectedMessage = '(No channels or focus points selected)';
        nothingFoundMessage = 'No valid fixtures found. Make sure all fixtures have channel,
… purpose, and focus data.';

    TYPE
        FOCUSPOINT = STRUCTURE
            name : STRING;
            x, y : REAL;
            include : BOOLEAN;
        END;

        FIXTUREINFO = STRUCTURE
            channel : STRING;
            purpose : STRING;
            focus : STRING;
            fx, fy : REAL;
            dx, dy : REAL;
            focusIndex : INTEGER;
            purposeIndex : INTEGER;
        END;

        PURPOSEINFO = STRUCTURE
            name : STRING;
            avgX : REAL;
            avgY : REAL;
            color : INTEGER;
            include : BOOLEAN;
            lowestChannel : INTEGER;
        END;

    VAR
        numFocusPoints, numFixtures, numPurposes : INTEGER;
        numHiddenPurposes, numHiddenFocusPoints : INTEGER;

        focusPoints : DYNARRAY [] OF FOCUSPOINT;
        fixtures : DYNARRAY [] OF FIXTUREINFO;
        purposes : DYNARRAY [] OF PURPOSEINFO;
        purposesCopy : DYNARRAY [] OF PURPOSEINFO;
        hiddenPurposes, hiddenFocusPoints : DYNARRAY [] OF STRING;

        rotation : REAL;
        originX, originY : REAL;

        leftColWidth, rightColWidth : INTEGER;

        dialogID, dialogResult : LONGINT;

        drawn : BOOLEAN;

        baseR, baseB, baseG : LONGINT;
        baseColor : INTEGER;

        resultStatus : BOOLEAN;
        objName : STRING;
        objHandle, recHd, wallHd : HANDLE;

        i, j : INTEGER;
        found : BOOLEAN;

    FUNCTION GetFieldVal(h : HANDLE; fieldName : STRING) : STRING;
        VAR
            recName : STRING;
            fieldVal : STRING;
        BEGIN
            recName := GetName(GetRecord(h, NumRecords(h)));
            fieldVal := GetRField(h, recName, fieldName);
            GetFieldVal := fieldVal;
```

```
        END;

    FUNCTION HueToColor(hue : REAL) : INTEGER;
        VAR
            r, g, b : LONGINT;
            result : INTEGER;
            total : LONGINT;
        BEGIN
            IF (hue <= 60) THEN BEGIN
                r := 65535;
                g := 65535 * hue/60;
                b := 0;
            END
            ELSE IF (hue <= 120) THEN BEGIN
                r := 65535 * (120-hue)/60;
                g := 65535;
                b := 0;
            END
            ELSE IF (hue <= 180) THEN BEGIN
                r := 0;
                g := 65535;
                b := 65535 * (hue-120)/60;
            END
            ELSE IF (hue <= 240) THEN BEGIN
                r := 0;
                g := 65535 * (240-hue)/60;
                b := 65535;
            END
            ELSE IF (hue <= 300) THEN BEGIN
                r := 65535 * (hue-240)/60;
                g := 0;
                b := 65535;
            END
            ELSE BEGIN
                r := 65535;
                g := 0;
                b := 65535 * (360-hue)/60;
            END;
            r := r * brightness;
            g := g * brightness;
            b := b * brightness;
            RGBToColorIndex(r, g, b, result);
            HueToColor := result;
        END;

    PROCEDURE GetHiddenValues;
        VAR
            result : STRING;
        BEGIN
            result := '_';
            WHILE (result <> '') DO BEGIN
                result := SubString(p__HiddenPurpose, delimiter, numHiddenPurposes+1);
                IF (result <> '') THEN BEGIN
                    numHiddenPurposes := numHiddenPurposes + 1;
                    ALLOCATE hiddenPurposes[1..numHiddenPurposes];
                    hiddenPurposes[numHiddenPurposes] := result;
                END;
            END;

            result := '_';
            WHILE (result <> '') DO BEGIN
                result := SubString(p__HiddenFocus, delimiter, numHiddenFocusPoints+1);
                IF (result <> '') THEN BEGIN
                    numHiddenFocusPoints := numHiddenFocusPoints + 1;
                    ALLOCATE hiddenFocusPoints[1..numHiddenFocusPoints];
                    hiddenFocusPoints[numHiddenFocusPoints] := result;
                END;
            END;
        END;

    PROCEDURE GrabFocusPoint(h : HANDLE);
        VAR
            newFocusPoint : FOCUSPOINT;
            xPos, yPos, zPos : REAL;
            include : BOOLEAN;
        BEGIN
```

```
152             newFocusPoint.name := GetFieldVal(h, 'Name');
153
154             Get3DCntr(h, xPos, yPos, zPos);
155             newFocusPoint.x := xPos;
156             newFocusPoint.y := yPos;
157
158             include := TRUE;
159             FOR i := 1 TO numHiddenFocusPoints DO BEGIN
160                 IF (hiddenFocusPoints[i] = newFocusPoint.name) THEN BEGIN
161                     include := FALSE;
162                 END;
163             END;
164             newFocusPoint.include := include;
165
166             numFocusPoints := numFocusPoints + 1;
167             ALLOCATE focusPoints[1..numFocusPoints];
168             focusPoints[numFocusPoints] := newFocusPoint;
169         END;
170
171     PROCEDURE GrabFixture(h : HANDLE);
172         VAR
173             newFixture : FIXTUREINFO;
174             xPos, yPos, zPos : REAL;
175         BEGIN
176             newFixture.channel := GetFieldVal(h, 'Channel');
177             newFixture.purpose := GetFieldVal(h, 'Purpose');
178             newFixture.focus := GetFieldVal(h, 'Focus');
179
180             Get3DCntr(h, xPos, yPos, zPos);
181
182             found := FALSE;
183             FOR i := 1 TO numFocusPoints DO BEGIN
184                 IF (newFixture.focus = focusPoints[i].name) THEN BEGIN
185                     newFixture.fx := focusPoints[i].x;
186                     newFixture.fy := focusPoints[i].y;
187                     newFixture.dx := xPos - focusPoints[i].x;
188                     newFixture.dy := yPos - focusPoints[i].y;
189                     newFixture.focusIndex := i;
190                     found := TRUE;
191                 END;
192             END;
193
194             IF ((found = TRUE) AND (newFixture.purpose <> '') AND (newFixture.channel <> '')) THEN
… BEGIN
195                 numFixtures := numFixtures + 1;
196                 ALLOCATE fixtures[1..numFixtures];
197                 fixtures[numFixtures] := newFixture;
198             END;
199         END;
200
201     PROCEDURE GrabPurposes;
202         VAR
203             newPurpose : PURPOSEINFO;
204             include : BOOLEAN;
205         BEGIN
206             FOR i := 1 TO numFixtures DO BEGIN
207                 found := FALSE;
208                 FOR j := 1 TO numPurposes DO BEGIN
209                     IF ((fixtures[i].purpose = purposes[j].name) AND NOT(found)) THEN BEGIN
210                         found := TRUE;
211                         purposes[j].avgX := purposes[j].avgX + fixtures[i].dx;
212                         purposes[j].avgY := purposes[j].avgY + fixtures[i].dy;
213                     END;
214                 END;
215                 IF NOT(found) THEN BEGIN
216                     numPurposes := numPurposes + 1;
217                     ALLOCATE purposes[1..numPurposes];
218                     newPurpose.name := fixtures[i].purpose;
219                     newPurpose.avgX := fixtures[i].dx;
220                     newPurpose.avgY := fixtures[i].dy;
221                     newPurpose.lowestChannel := Str2Num(fixtures[i].channel);
222
223                     include := TRUE;
224                     FOR j := 1 TO numHiddenPurposes DO BEGIN
225                         IF (hiddenPurposes[j] = newPurpose.name) THEN BEGIN
226                             include := FALSE;
```

```
227                         END;
228                     END;
229                     newPurpose.include := include;
230                     purposes[numPurposes] := newPurpose;
231                 END;
232             END;
233
234             IF (numPurposes > 0) THEN
235                 SortArray(purposes, numPurposes, 1);
236
237             FOR i := 1 TO numFixtures DO BEGIN
238                 found := FALSE;
239                 FOR j := 1 TO numPurposes DO BEGIN
240                     IF ((fixtures[i].purpose = purposes[j].name) AND NOT (found)) THEN BEGIN
241                         found := TRUE;
242                         fixtures[i].purposeIndex := j;
243                     END;
244                 END;
245             END;
246
247             FOR i := 1 TO numFixtures DO BEGIN
248                 IF (Str2Num(fixtures[i].channel) <
… purposes[fixtures[i].purposeIndex].lowestChannel) THEN
249                     purposes[fixtures[i].purposeIndex].lowestChannel :=
… Str2Num(fixtures[i].channel);
250                 END;
251         END;
252
253     PROCEDURE ScaleVectors;
254         VAR
255             total : REAL;
256         BEGIN
257             FOR i := 1 TO numPurposes DO BEGIN
258                 total := Sqrt(Sqr(purposes[i].avgX) + Sqr(purposes[i].avgY));
259                 purposes[i].avgX := purposes[i].avgX / total * pOffsetDistance;
260                 purposes[i].avgY := purposes[i].avgY / total * pOffsetDistance;
261             END;
262         END;
263
264     PROCEDURE AssignColors;
265         VAR
266             numIncluded, numSeen : INTEGER;
267             hue : REAL;
268             lowest, previousLowest : INTEGER;
269             savedIndex : INTEGER;
270             roundedNumIncluded : INTEGER;
271         BEGIN
272             numIncluded := 0;
273             FOR i := 1 TO numPurposes DO BEGIN
274                 IF (purposes[i].include = TRUE) THEN
275                     numIncluded := numIncluded + 1;
276             END;
277             IF (numIncluded <= 3) THEN
278                 roundedNumIncluded := 3
279             ELSE IF (numIncluded <= 6) THEN
280                 roundedNumIncluded := 6
281             ELSE
282                 roundedNumIncluded := numIncluded;
283
284             previousLowest := -1;
285             FOR i := 1 TO numIncluded DO BEGIN
286                 lowest := 30000;
287                 FOR j := 1 TO numPurposes DO BEGIN
288                     IF ((purposes[j].include = TRUE) AND (purposes[j].lowestChannel < lowest) AND
… (purposes[j].lowestChannel > previousLowest)) THEN BEGIN
289                         lowest := purposes[j].lowestChannel;
290                         savedIndex := j;
291                     END;
292                 END;
293                 hue := (i-1) / roundedNumIncluded * 360;
294                 purposes[savedIndex].color := HueToColor(hue);
295                 previousLowest := lowest;
296             END;
297         END;
298
299     PROCEDURE DrawChannels;
```

```
300         VAR
301             drawX, drawY : REAL;
302             purposeIndex, focusIndex : INTEGER;
303         BEGIN
304             TextFont(GetFontID('Arial'));
305             TextSize(pChannelTextSize);
306             TextJust(2);
307             TextVerticalAlign(3);
308             IF pFillText THEN FillPat(1) ELSE FillPat(0);
309             PenFore(baseColor);
310             TextFace([]);
311 
312             FOR i := 1 TO numFixtures DO BEGIN
313                 purposeIndex := fixtures[i].purposeIndex;
314                 focusIndex := fixtures[i].focusIndex;
315                 IF ((purposes[purposeIndex].include = TRUE) AND (focusPoints[focusIndex].include =
    … TRUE)) THEN BEGIN
316                     IF (pColorChannels) THEN
317                         PenFore(purposes[purposeIndex].color);
318                     IF (pLockLocation) THEN BEGIN
319                         drawX := fixtures[i].fx + purposes[purposeIndex].avgX - originX;
320                         drawY := fixtures[i].fy + purposes[purposeIndex].avgy - originY;
321                     END
322                     ELSE BEGIN
323                         drawX := fixtures[i].fx + purposes[purposeIndex].avgX;
324                         drawY := fixtures[i].fy + purposes[purposeIndex].avgy;
325                     END;
326                     TextOrigin(drawX, drawY);
327                     CreateText(fixtures[i].channel);
328                     drawn := TRUE;
329                 END;
330             END;
331         END;
332 
333     PROCEDURE DrawFocusPoints;
334         BEGIN
335             TextFace([Italic]);
336             TextSize(pFocusTextSize);
337             PenFore(baseColor);
338 
339             FOR i := 1 TO numFocusPoints DO BEGIN
340                 IF (focusPoints[i].include = TRUE) THEN BEGIN
341                     IF (pLockLocation) THEN
342                         TextOrigin(focusPoints[i].x - originX, focusPoints[i].y - originY)
343                     ELSE
344                         TextOrigin(focusPoints[i].x, focusPoints[i].y);
345                     CreateText(focusPoints[i].name);
346                     drawn := TRUE;
347                 END;
348             END;
349         END;
350 
351     PROCEDURE SetColWidths;
352         BEGIN
353             leftColWidth := minLeftColWidth;
354             rightColWidth := minRightColWidth;
355             FOR i := 1 TO numPurposes DO BEGIN
356                 IF (GetDlgCtrlWidthStdCh(purposes[i].name) > leftColWidth) THEN
357                     leftColWidth := GetDlgCtrlWidthStdCh(purposes[i].name);
358             END;
359             FOR i := 1 TO numFocusPoints DO BEGIN
360                 IF (GetDlgCtrlWidthStdCh(focusPoints[i].name) > rightColWidth) THEN
361                     rightColWidth := GetDlgCtrlWidthStdCh(focusPoints[i].name);
362             END;
363         END;
364 
365     FUNCTION CreateDialog : LONGINT;
366         VAR
367             refIndex : INTEGER;
368         BEGIN
369             dialogID := CreateLayout('Channel Marker', FALSE, 'Apply', 'Cancel');
370 
371             IF (numFixtures > 0) THEN BEGIN
372                 CreateStaticText(dialogID, 11, 'Purpose:', 9);
373                 CreateStaticText(dialogID, 12, 'Area:', 6);
374                 SetFirstLayoutItem(dialogID, 11);
```

```
375                 SetStaticTextStyle(dialogID, 11, 1);
376                 SetStaticTextStyle(dialogID, 12, 1);
377
378                 FOR i := 1 TO numPurposes DO BEGIN
379                     CreateStaticText(dialogID, 20+i*4, purposes[i].name, leftColWidth);
380                     CreateCheckBox(dialogID, 20+i*4+1, '');
381                 END;
382                 IF (numPurposes > 0) THEN BEGIN
383                     SetBelowItem(dialogID, 11, 24, 0, 1);
384                     SetRightItem(dialogID, 24, 25, 0, 0);
385                     FOR i := 2 TO numPurposes DO BEGIN
386                         IF (((i-1) MOD maxColCount) = 0) THEN
387                             SetRightItem(dialogID, 20+(i-maxColCount)*4+1, 20+i*4, 0, 0)
388                         ELSE
389                             SetBelowItem(dialogID, 20+i*4-4, 20+i*4, 0, 0);
390                         SetRightItem(dialogID, 20+i*4, 20+i*4+1, 0, 0);
391                     END;
392                     refIndex := (((i-1) DIV maxColCount)*15+1);
393                     SetRightItem(dialogID, 20+refIndex*4+1, 12, interColWidth, -7);
394                 END;
395
396                 FOR i := 1 TO numFocusPoints DO BEGIN
397                     CreateStaticText(dialogID, 20+i*4+2, focusPoints[i].name, rightColWidth);
398                     CreateCheckBox(dialogID, 20+i*4+3, '');
399                 END;
400                 IF (numFocusPoints > 0) THEN BEGIN
401                     SetBelowItem(dialogID, 12, 26, 0, 1);
402                     SetRightItem(dialogID, 26, 27, 0, 0);
403                     FOR i := 2 TO numFocusPoints DO BEGIN
404                         IF (((i-1) MOD maxColCount) = 0) THEN
405                             SetRightItem(dialogID, 20+(i-maxColCount)*4+3, 20+i*4+2, 0, 0)
406                         ELSE
407                             SetBelowItem(dialogID, 20+i*4+2-4, 20+i*4+2, 0, 0);
408                         SetRightItem(dialogID, 20+i*4+2, 20+i*4+3, 0, 0);
409                     END;
410                 END;
411
412                 CreatePushButton(dialogID, 13, 'Show All');
413                 CreatePushButton(dialogID, 14, 'Hide All');
414                 SetBelowItem(dialogID, 20+numPurposes*4, 13, 0, 1);
415                 SetBelowItem(dialogID, 13, 14, 0, -2);
416                 CreatePushButton(dialogID, 15, 'Show All');
417                 CreatePushButton(dialogID, 16, 'Hide All');
418                 SetBelowItem(dialogID, 20+numFocusPoints*4+2, 15, 0, 1);
419                 SetBelowItem(dialogID, 15, 16, 0, -2);
420             END
421             ELSE BEGIN
422                 CreateStaticText(dialogID, 11, nothingFoundMessage, -1);
423                 SetFirstLayoutItem(dialogID, 11);
424             END;
425
426             CreateDialog := dialogID;
427         END;
428
429     PROCEDURE HandleDialog(VAR item : LONGINT; data : LONGINT);
430         VAR
431             checked : BOOLEAN;
432             hiddenString : STRING;
433             first : BOOLEAN;
434         BEGIN
435             CASE item OF
436                 SetupDialogC: BEGIN
437                     FOR i := 1 TO numPurposes DO BEGIN
438                         SetBooleanItem(dialogID, 20+i*4+1, purposes[i].include);
439                     END;
440                     FOR i := 1 TO numFocusPoints DO BEGIN
441                         SetBooleanItem(dialogID, 20+i*4+3, focusPoints[i].include);
442                     END;
443                 END;
444                 1: BEGIN
445                     first := TRUE;
446                     hiddenString := '';
447                     FOR i := 1 TO numPurposes DO BEGIN
448                         GetBooleanItem(dialogID, 20+i*4+1, checked);
449                         purposes[i].include := checked;
450                         IF NOT(checked) THEN BEGIN
```

```
451                             IF (first = TRUE) THEN BEGIN
452                                 hiddenString := purposes[i].name;
453                                 first := FALSE;
454                             END
455                             ELSE
456                                 hiddenString := Concat(hiddenString, delimiter, purposes[i].name);
457                         END;
458                     END;
459                     SetRField(objHandle, GetName(recHd), '__HiddenPurpose', hiddenString);
460 
461                     first := TRUE;
462                     hiddenString := '';
463                     FOR i := 1 TO numFocusPoints DO BEGIN
464                         GetBooleanItem(dialogID, 20+i*4+3, checked);
465                         focusPoints[i].include := checked;
466                         IF NOT(checked) THEN BEGIN
467                             IF (first = TRUE) THEN BEGIN
468                                 hiddenString := focusPoints[i].name;
469                                 first := FALSE;
470                             END
471                             ELSE
472                                 hiddenString := Concat(hiddenString, delimiter,
…  focusPoints[i].name);
473                         END;
474                     END;
475                     SetRField(objHandle, GetName(recHd), '__HiddenFocus', hiddenString);
476 
477                 END;
478                 13: BEGIN
479                     FOR i := 1 TO numPurposes DO BEGIN
480                         SetBooleanItem(dialogID, 20+i*4+1, TRUE);
481                     END;
482                 END;
483                 14: BEGIN
484                     FOR i := 1 TO numPurposes DO BEGIN
485                         SetBooleanItem(dialogID, 20+i*4+1, FALSE);
486                     END;
487                 END;
488                 15: BEGIN
489                     FOR i := 1 TO numFocusPoints DO BEGIN
490                         SetBooleanItem(dialogID, 20+i*4+3, TRUE);
491                     END;
492                 END;
493                 16: BEGIN
494                     FOR i := 1 TO numFocusPoints DO BEGIN
495                         SetBooleanItem(dialogID, 20+i*4+3, FALSE);
496                     END;
497                 END;
498             END;
499         END;
500 
501     PROCEDURE DrawWarning;
502         VAR
503             centerX, centerY : REAL;
504         BEGIN
505             centerX := 0;
506             centerY := 0;
507             FOR i := 1 TO numFocusPoints DO BEGIN
508                 centerX := centerX + focusPoints[i].x;
509                 centerY := centerY + focusPoints[i].y;
510             END;
511             centerX := centerX / numFocusPoints;
512             centerY := centerY / numFocusPoints;
513 
514             TextSize(16);
515             FillPat(1);
516             TextFace([Bold]);
517             IF (pLockLocation) THEN
518                 TextOrigin(centerX-originX, centerY-originY)
519             ELSE
520                 TextOrigin(centerX, centerY);
521             CreateText(nothingSelectedMessage);
522         END;
523 
524     BEGIN
525         resultStatus := GetCustomObjectInfo(objName, objHandle, recHd, wallHd);
```

```
526         SetObjectVariableBoolean(objHandle, 800, TRUE);
527
528         GetSymLoc(objHandle, originX, originY);
529         rotation := GetSymRot(objHandle);
530         HRotate(objHandle, originX, originY, -rotation);
531
532         IF pUpdateData THEN SetRField(objHandle, GetName(recHd), 'UpdateData', 'FALSE');
533
534         numFocusPoints := 0;
535         numFixtures := 0;
536         numPurposes := 0;
537         numHiddenPurposes := 0;
538         numHiddenFocusPoints := 0;
539
540         GetHiddenValues;
541
542         ForEachObject(GrabFocusPoint, (PON='Focus Point Object'));
543         IF (numFocusPoints > 0) THEN
544             SortArray(focusPoints, numFocusPoints, 1);
545
546         ForEachObject(GrabFixture, (PON='Lighting Device'));
547
548         GrabPurposes;
549
550         IF ((pOpenEditor = TRUE) OR (GetRField(objHandle, GetName(recHd), '__Seen') = 'Second'))
… THEN BEGIN
551             SetRField(objHandle, GetName(recHd), '__Seen', 'Done');
552             SetColWidths;
553             dialogID := CreateDialog;
554             dialogResult := RunLayoutDialog(dialogID, HandleDialog);
555
556             SetRField(objHandle, GetName(recHd), 'OpenEditor', 'FALSE');
557         END;
558         IF (GetRField(objHandle, GetName(recHd), '__Seen') = 'First') THEN
559             SetRField(objHandle, GetName(recHd), '__Seen', 'Second');
560
561         ScaleVectors;
562
563         IF (pColorChannels = TRUE) THEN
564             AssignColors;
565
566         drawn := FALSE;
567         GetPenFore(objHandle, baseR, baseG, baseB);
568         RGBToColorIndex(baseR, baseG, baseB, baseColor);
569         DrawChannels;
570         IF (pShowFocus = TRUE) THEN
571             DrawFocusPoints;
572
573         IF (((numFixtures < 1) AND (pShowFocus = FALSE)) OR (numFocusPoints < 1) OR (drawn =
… FALSE)) THEN
574             DrawWarning;
575     END;
576
577 Run(ChannelMarker);
```