



# AS/400教材

海辉软件（大连）有限公司

二零零八年十月



## 前 言

## 目 录

第一章 AS/400 系统概况 .....	- 1 -
1.1 AS/400 发展历史 .....	- 1 -
1.2 iSeries 400 的特点 .....	- 1 -
1.3 iSeries 400 集成的工具 .....	- 2 -
1.4 iSeries 400 系统的基本概念 .....	- 2 -
第二章 仿真模拟器与程序设计开发管理工具 .....	- 4 -
2.1 PCOMM 5.0 安装与配置 .....	- 4 -
2.2 程序设计开发管理工具 .....	- 5 -
2.2.1 PDM 简介 .....	- 5 -
2.2.2 启动 PDM .....	- 6 -
2.2.3 处理库 .....	- 7 -
2.2.4 处理对象 .....	- 8 -
2.2.5 处理成员 .....	- 9 -
2.2.6 处理用户定义的选项 .....	- 10 -
第三章 数据库 .....	- 24 -
3.1 数据库文件基本概念 .....	- 24 -
3.1.1 数据库文件定义方法 .....	- 24 -
3.1.2 物理文件: .....	- 25 -
3.1.3 逻辑文件: .....	- 25 -
3.2 建立物理文件 .....	- 25 -
3.2.1 典型物理文件程序范例 .....	- 25 -
3.2.2 物理文件的代码范例 .....	- 26 -
3.2.3 DDS 规则讲解 .....	- 28 -
3.2.4 生成物理文件 .....	- 31 -
3.3 建立逻辑文件 .....	- 34 -
3.3.1 逻辑文件种类 .....	- 34 -
3.3.2 简单逻辑文件程序范例 .....	- 34 -
3.3.3 逻辑文件的编码范例 .....	- 36 -
3.3.4 建立逻辑文件 .....	- 40 -
3.3.5 其他种类逻辑文件介绍 .....	- 41 -
3.4 数据文件实用程序 (DFU) 使用与说明 .....	- 42 -
3.4.1 DFU 简介 .....	- 42 -
3.4.2 运行 DFU .....	- 43 -
3.4.3 创建 DFU .....	- 44 -
3.4.4 运行临时 DFU 程序 .....	- 49 -

第四章 控制语言 (CL) .....	50 -
4.1 基本概念.....	50 -
4.2 CL 编程 .....	50 -
4.2.1 程序的开始与结束.....	50 -
4.2.2 变量及其定义.....	51 -
4.2.3 CL 常用命令 .....	51 -
4.2.4 CL 过程的组成部分 .....	55 -
4.2.5 CL 过程中使用的命令分组说明 .....	56 -
4.2.6 变量的使用.....	58 -
4.2.6.1 变量说明.....	58 -
4.2.6.2 变量值中小写字符的限制.....	58 -
4.2.6.3 变量赋值.....	58 -
4.2.6.4 在 CL 过程中写注释.....	58 -
4.2.7 CL 程序内部的逻辑控制 .....	59 -
4.2.7.1 使用 GOTO 命令及标号.....	60 -
4.2.7.2 使用 IF 命令.....	60 -
4.2.7.3 使用 DO 命令和 DO 组.....	62 -
4.2.7.4 使用 ELSE 命令.....	63 -
4.2.7.5 使用嵌套的 IF 命令.....	65 -
4.2.7.6 使用*AND、*OR、和*NOT 操作 .....	67 -
4.2.8 系统资源的获取.....	70 -
4.2.8.1 日期格式转换命令: CVTDAT.....	70 -
4.2.8.2 检索系统值命令: RTVSYSVAL.....	70 -
4.2.8.3 检索配置源命令: RTVCFGSRC.....	70 -
4.2.8.4 配置状态检索命令: RTVCFGSTS.....	70 -
4.2.8.5 检索网络属性的命令: RTVNETA.....	70 -
4.2.8.6 检索作业属性命令: RTVJOBA.....	70 -
4.2.8.7 检索对象描述命令: RTVOBJD.....	70 -
4.2.8.8 检索用户档案命令: RTVUSRPRF.....	70 -
4.3 程序间的通讯.....	71 -
4.3.1 CALL 命令的使用 .....	71 -
4.3.2 数据队列的程序通讯.....	71 -
4.3.2.1 数据队列的优点.....	71 -
4.3.2.2 数据队列的发送.....	72 -
4.3.2.3 数据队列的接收.....	72 -
4.3.2.4 数据队列的清除.....	72 -
4.3.3 数据域的程序通讯.....	73 -

4.4 测试功能.....	73 -
4.4.1 CL 程序的编译 .....	73 -
4.4.2 CL 程序的测试 .....	74 -
第五章 显示文件.....	75 -
5.1 显示文件基本概念.....	75 -
5.2 用 SEU 建立显示文件.....	75 -
5.2.1 显示文件程序范例.....	75 -
5.2.2 显示文件程序编写.....	77 -
5.2.3 显示文件 DDS 规则讲解.....	78 -
5.2.4 编译生成对象文件.....	86 -
5.2.5 显示子文件介绍.....	86 -
5.3 屏幕设计辅助工具（SDA）使用与说明.....	86 -
5.3.1 SDA 简介 .....	86 -
5.3.2 SDA 的启动 .....	87 -
5.3.3 通过 SDA 创建显示文件.....	88 -
第六章 帐票文件（RLU）.....	104 -
6.1 基本概念.....	104 -
6.2 举例说明 RLU 生成报表的过程.....	104 -
6.2.1 PF 文件说明： .....	104 -
6.2.2 使用 RLU 生成报表的具体操作步骤： .....	105 -
6.2.2.1 为打印文件选择数据库： .....	105 -
6.2.2.2 定义记录格式： .....	108 -
6.2.2.3 放置数据库字段： .....	110 -
6.2.2.4 定义表头： .....	114 -
6.2.2.5 合并记录格式.....	122 -
6.2.2.6 建立变量型字段.....	124 -
6.2.2.7 保存报表设计名生成打印文件 PRTF .....	127 -
6.2.3 报表 PRTF 文件的源码.....	131 -
第七章 RPG 语言 .....	132 -
7.1 基本概念.....	132 -
7.2 最简单的 RPGLE 程序 .....	132 -
7.2.1 举例准备.....	133 -
7.2.2 简单的程序流程.....	135 -
7.2.3 常见的程序流程.....	136 -
7.3 程序代码行的编写.....	137 -
7.3.1 F 行说明.....	137 -
7.3.2 D 行说明.....	141 -

7.3.3 入口参数.....	- 146 -
7.3.4 C 行说明.....	- 148 -
7.3.5 ILE 操作码.....	- 151 -
7.3.5.1 A—C 开头 .....	- 151 -
7.3.5.2 D--E 开头 .....	- 159 -
7.3.5.3 F--N 开头 .....	- 164 -
7.3.5.4 O--R 开头 .....	- 171 -
7.3.5.5 S--Z 开头 .....	- 175 -
第八章 编辑错误处理与 DEBUG 调试.....	- 182 -
8.1 编辑错误处理.....	- 182 -
8.1.1 编译清单中的结构化操作缩进.....	- 182 -
8.1.2 获取编译清单.....	- 183 -
8.1.3 改正编译错误.....	- 184 -
8.1.4 常见错误信息.....	- 185 -
8.2 DEBUG 调试.....	- 186 -
8.2.1 调试时的入口参数问题.....	- 186 -
8.2.2 RPG 的调试 .....	- 186 -
8.2.3 RPGLE 的调试 .....	- 189 -
8.2.4 批处理程序的调试.....	- 190 -
第九章 数据库操作.....	- 196 -
9.1 Query.....	- 196 -
9.1.1 查询的基本概念.....	- 196 -
9.1.2 使用查询命令.....	- 196 -
9.2 SQL 在 AS400 上的应用实例 .....	- 201 -
9.2.1 SQL 工具的启动 .....	- 201 -
9.2.2 SQL 使用举例 .....	- 202 -
9.3 事务处理 COMMIT .....	- 203 -
9.4 关于锁表的问题 LOCK .....	- 204 -
9.5 SAVF 的备份与恢复 .....	- 205 -
9.6 数据导入导出图解.....	- 206 -
9.6.1 数据导出.....	- 206 -
9.6.2 数据导入.....	- 209 -
第十章 附    录.....	- 211 -
10.1 CL 程序实例: .....	- 211 -
10.1.1 起动所用的初始程序（程序员）.....	- 211 -
10.1.2 把对象从测试库移到产品库中.....	- 211 -
10.1.3 在应用程序中保存规定的对象.....	- 212 -



10.1.4 提交作业（系统操作员） .....	- 212 -
10.1.5 打印程序输出 .....	- 213 -
10.1.6 在 QTEMP 下生成临时文件 .....	- 213 -
10.2 AS/400 常用 CL 命令表 .....	- 214 -
10.3 RPG 程序实例： .....	- 218 -
10.3.1 查询（指示画面+SFL+报表） .....	- 218 -
10.4 RPG 编程常用操作码详解 .....	- 233 -
10.5 RPG 编程常用数组命令 .....	- 235 -
10.6 Query/400 创建高级查询 .....	- 238 -





## 第一章 AS/400 系统概况

AS/400 是当今世界上最流行的中小型、多用户商业计算机系统，在多用户服务器领域里，始终保持着最畅销的地位。目前 AS/400 在全球的装机量已超过 75 万套，覆盖 150 多个国家，支持 40 多种语言，有近 1 万个商业伙伴和独立软件商，3 万多个商业应用。广泛应用于流通、金融证券、制造、运输等各个行业。

AS/400e 及其之后的产品系列，融合了 Java、Domino、服务器整合与逻辑分区、Where sphere 和商业智能等许多业界最新技术，并增加了对欧元的支持，帮助用户更有效地将企业产品和服务推向市场，在新兴的电子商务领域获益。

本章主要介绍了 AS/400 的发展历史、iSeries 400 的特点、iSeries 400 的集成工具。同时也对 AS/400 的基本概念做了介绍。

### 1.1 AS/400 发展历史

为了深入了解一个系统，必须熟悉它发展的历史和背景。多数的计算机系统是从原有的系统基础上发展起来的。AS/400 的发展历史：

- ◇ 1969 年 6 月发布 System/3 (Batch machine)
- ◇ 1975 年 1 月发布 System/32 (用于小型商务办公环境)
- ◇ 1977 年 4 月发布 System/34
- ◇ 1978 年 10 月发布 System/38 (第一代 AS/400, 商业上失败, 技术上成功)
- ◇ 1983 年 5 月发布 System/36 (商业上非常成功)
- ◇ 1988 年 6 月发布 AS/400 B 型号 (Application System/400, 第二代 AS/400)
- ◇ 1994 年 5 月发布 AS/400 先进系列/服务器 (Advanced System/400, 第三代 AS/400)
- ◇ 1995 年 6 月发布 AS/400 Power PC 先进系列/服务器
- ◇ 1997 年 8 月发布 AS/400e 系列
- ◇ 1999 年 2 月发布 AS/400 e 服务器 170、7XX
- ◇ 1999 年 7 月发布 AS/400e Domino 专用服务器 (莲花宝箱)
- ◇ 2000 年 10 月发布 iSeries 400, 包括 270、820、830、840 及 IBM 莲花宝箱 (DSD)

### 1.2 iSeries 400 的特点

iSeries 400 通过紧密集成硬件、软件、中间件和操作系统提供能够满足不同业务需求的高性能、可靠和易于使用特性。从 e 系列开始，在电子商务大舞台上开始扮演更为重要的角色。它可以提供扩展业务确保电子商务优势所需的技术和工具。它的突出特点有：

1. 卓越的性能，不断获得各种荣誉和认可，全面实现 64 位处理，先进的体系结构，最领先的 SOI 芯片技术；
2. 卓越的可靠性和可用性，系统可提供 99.97% 的可用性，使 iSeries 成为商业关键事务处理的首选服务器平台；
3. 高度集成的系统环境，使用户不需要太多的系统管理员就能保证其应用平稳地运行；

4. 严密的安全防范系统，荣获美国联邦政府定义的商业计算机最高安全性级别 C2 级认证，堪称世界上“最安全”的计算机系统。从来没有发现过病毒。
5. 开放标准兼容性，使用户把运行在不同平台上的业务系统和数据平滑地连接在一起；
6. 简单易用，使用户（尤其是新用户）很快就能掌握它的系统管理和应用程序开发。
7. 允许在一台服务器上安装 Linux、Java、Windows2000 和 Domino 等多种应用。
8. 在系统中可以配备 16 台 PC 服务器 Netfinity，可以运行多种不同的操作系统，并共享主机系统的资源。
9. 是唯一一种能直接（固有）支持多种不同文件结构，如 PC 文件、Unix 文件、Netware 文件、Domino 文件、ASCII 文件、EDBCID 文件的系统。

### 1.3 iSeries 400 集成的工具

iSeries 400 为程序员提供了丰富而强大的工具，主要有以下：

1. PDM (PROGRAMMING DEVELOPMENT MANAGER)：可以用来处理源代码、对象和库。为程序员建立源文件成员、访问 SEU 和许多其他有用的工具提供方便。
2. SEU (SOURCE ENTRY UTILITY)：是一个全屏编辑工具，可以建立和编辑源文件成员，当启动时，能够输入新的源语句，修改、删除、复制、移动已存在的源语句，具有语言相关提示和语法检查功能，且具有分屏编辑/浏览功能。
3. SDA (SCREEN DESIGN AID)：可用来交互式设计、创建和维护应用屏幕，包括显示文件和菜单，且可以将用户设计的屏幕规范地自动转换成 DDS 源代码，简化了菜单和显示文件的创建。
4. RLU (REPORT LAYOUT UTILITY)：可用来交互式定义打印报表的格式分布，建立打印文件，且可以将用户设计的报表格式分布规范地自动转换成 DDS 源代码，简化了报表的设计和修改，使用它可以在屏幕上直观地设计打印报表。
5. DFU (DATA FILE UTILITY)：能够快速定义、创建面向数据录入、查询或文件维护的 DFU 程序，而不需要编程。对开发应用建立测试数据库尤其有用。
6. QUERY/400：QUERY/400 特许程序是一个非常有用和容易使用的决策支持工具，可用来获取外部描述数据库文件信息。它允许使用单个文件或联结最多 32 个不同文件的数据，产生的报表可以打印、屏幕显示或存放在新的数据库文件中，功能强大。

### 1.4 iSeries 400 系统的基本概念

数据库：从 AS/400 开始，关系数据库就一直是 OS/400 操作系统的一部分。这个集成的、完整的数据库在 1999 年 2 月被标上了“DB2 UDB for iSeries”品牌。尽管与其它 DB2 平台一样，该产品的每个新版本都会不断添加新的功能和特性，但是最初的数据库引擎仍然是相同的。因为 DB2 UDB for iSeries 是 OS/400 操作系统的一部分，所以数据库系统的版本及发行版与操作系统使用的版本及发行版是一样的

库：一个库用来把相关对象组成一组，且由名字来查找使用的对象。这样一个库是一组对象的目录。也可用库来把对象组成不同意义上的集合，例如，可以根据安全需要、备份、或处理来把对象分组。一个库中有多少对象，系统中能有多少个库仅由存储总量来限定，但不能多于 8000 以便保证完成保存操作。

**对象：**对象是 AS/400 系统中命令执行操作的基本工作单元。对象有很多类型。例如，库的类型为\*LIB，文件的类型为\*FILE，程序的类型是\*PGM。对象也有属性，它是类型的一部分，描述对象的特性。例如，\*PGM 类型的属性可以是 RPG，它包括用 RPG 源码生成的程序；\*FILE 类型的属性可以是 DSPF，说明它是显示文件。

**成员：**成员是物理文件（PF-SRC 或 PF-DTA）记录的子集。每个成员都要符合文件的特性，可用 PDM 命令定义或选择成员的类型。当是 PF-SRC 下成员时，又叫源程序，通过编译，可以生成对应的对象。

**文件：**文件是类型为\*FILE 及其属性的对象。例如，源物理文件属性为 PF-SRC；数据物理文件属性为 PF-DTA；数据逻辑文件属性为 LF；打印文件属性为 PRTF；显示文件属性为 DSPF。物他们都可以存在对应的描述成员（源程序）。

**库列表：**对象名可以是限定名也可以是仅对象名本身。如果用限定名，系统要在规定的库中找对象，如果仅用对象名，系统要检索库列表，直到找到此对象的第一次出现或检索完库列表中的所有库却没有找到给定的对象。要检索的库以及它们被检索的顺序，是由库列表决定的，系统在作业启动时为它建立一个初始库列表。下面给出库列表的检索顺序：

库列表检索顺序

QSYS QUSRSYS QHCPSYS	系统部分
QPDA	产品库 1
QRPGR	产品库 2
OELIB	当前库
QELIB QGPL QTEMP	用户部分

一个库列表有四部分。第一部分是系统部分，它是最早检索的一些库。这部分规定了所有在系统中运行的作业使用的库。在安装系统时，这部分由 QSYS、QUSRSYS、QHLPSYS 和 QSYS2 组成。第二部分是产品库。它是由系统在用户运行命令或菜单时修改的。根据完成功能的不同，产品库在作业运行时也各不相同。第三部分是当前库。它是建立对象时所用的缺省库。用户可用 CL 命令规定作业的当前库。第四部分是用户部分。它包括应用程序完成功能所使用的库。当安装系统时，它包括 QGPL、QTEMP，每个作业都有自己的 QTEMP，其它作业看不到，当作业结束时，QTEMP 库也随之消失。

## 第二章 仿真模拟器与程序设计开发管理工具

### 2.1 PCOMM 5.0 安装与配置

PCOMM 是常用的 AS/400 仿真终端软件，在本节主要讲解 PCOMM 5.0 的安装与配置。点击执行安装文件，全部采用默认设置，点击下一步，直到安装完毕即可。点击配置图标，主机类型选择 AS/400，接口选择 LAN，连接选择 TCP/IP，设定如图：



点击链路参数项，如图配置。如本例 AS/400 服务器 IP 地址为 192.168.101.20，则如下图填写，端口默认 23 即可。



点击会话参数，工作站配置如下：



**配置说明：**

屏幕大小：根据选择尺寸不同，画面显示像素不同。根据个人习惯选择。

会话类型：分为显示和打印两种类型。

字符码主机代码页：根据语言，服务器不同而不同。一般由项目规定。如果选择不当，源程序中会出现显示乱码问题。如果没有特殊要求，日文操作系统默认 930 码。中文操作系统按默认 1388 码即可。

工作站 ID：默认不填则系统会自动生成，一般为了安全和便于管理，AS/400 管理员会分配给用户固定的 ID。

设定好后，点击确定，短暂的连接后，显示登录画面。

Sign On

System . . . . .	:	XXXXXXXX
Subsystem . . . . .	:	QINTER
Display . . . . .	:	DMK079

User . . . . .	_____
Password . . . . .	_____
Program/procedure . . . . .	_____
Menu . . . . .	_____
Current library . . . . .	_____

到此，仿真终端 PCOMM 就配置完成了。

## 2.2 程序设计开发管理工具

### 2.2.1 PDM 简介

PDM 全称 PROGRAMMING DEVELOPMENT MANAGER，可以用来处理源代码、对象和库。为程序员建立源文件成员、访问 SEU 和许多其他有用的工具提供方便。是我们日常开发必须用到的工具，做为一个 RPG 程序员，不能不掌握它。

在登录画面，根据管理员分配的用户名，密码，键入正确即可进入主菜单。如下图：

MAIN	OS/400 Main Menu	System: XXXXXXXX
Select one of the following:		
<ul style="list-style-type: none"><li>1. User tasks</li><li>2. Office tasks</li><li>3. General system tasks</li><li>4. Files, libraries, and folders</li><li>5. Programming</li><li>6. Communications</li><li>7. Define or change the system</li><li>8. Problem handling</li><li>9. Display a menu</li><li>10. Information Assistant options</li><li>11. Client Access/400 tasks</li></ul>		
90. Sign off		
Selection or command		
====>		
<hr/>		
F3=Exit F4=Prompt F9=Retrieve F12=Cancel F13=Information Assistant		
F23=Set initial menu		
(C) COPYRIGHT IBM CORP. 1980, 2002.		

在本章，主要介绍项目开发平台环境 PDM 及源语句编辑工具 SEU 的使用。

### 2.2.2 启动 PDM

启动 PDM 的方法有很多。主菜单或在命令行上用 STRPDM 命令启动 PDM。根据处理的对象不同，也可用下列命令启动 PDM：WRKLBPDM，WRKOBJPDM，WRKMBRPDM。本章主要讲解 STRPDM 启动 PDM 的方法。在主菜单命令行键入 STRPDM 回车，启动 PDM。

MAIN	OS/400 Main Menu	System: XXXXXXXX
Select one of the following:		
<ul style="list-style-type: none"><li>1. User tasks</li><li>2. Office tasks</li><li>3. General system tasks</li><li>4. Files, libraries, and folders</li><li>5. Programming</li><li>6. Communications</li><li>7. Define or change the system</li><li>8. Problem handling</li><li>9. Display a menu</li><li>10. Information Assistant options</li><li>11. Client Access/400 tasks</li></ul>		
90. Sign off		
Selection or command		
====>STRPDM		
<hr/>		
F3=Exit F4=Prompt F9=Retrieve F12=Cancel F13=Information Assistant		
F23=Set initial menu		
(C) COPYRIGHT IBM CORP. 1980, 2002.		

下图就是 PDM 的应用界面。

AS/400 程序设计开发管理程序 (PDM)

选择下列其中一项：

- 1. 使用库
- 2. 使用对象
- 3. 使用成员
- 9. 使用用户定义选项

选择或命令  
===> \_\_\_\_\_

---

F3= 退出	F4= 提示	F9= 检索	F10= 命令输入
F12= 取消	F18= 更改缺省值		

(C) COPYRIGHT IBM CORP. 1981, 2002.

后面的小结，将根据选择命令顺序说明 PDM 使用方法。

### 2.2.3 处理库

在 PDM 主菜单命令行键入 1 回车，进入处理库指定菜单。键入要处理的库，执行即可进入根据指定参数列出的库清单界面。如果要查看 AS/400 中现有的所有库，可以\*ALL 查看。如下图：

使用 PDM 处理库				XXXXXXXX
列表类型 . . . . .	<u>*ALL</u>	定位至 . . . . .		
输入选项，按“ 执行” 键。				
2= 更改	3= 复制	4= 删除	5= 显示	
7= 重命名	8= 显示说明	9= 保存	10= 复原 ...	
Opt 库	类型	文本		
___ TST010	*PROD	RPG STUDY LIB		
				底部
参数或命令				
====>				
F3= 退出	F4= 提示	F5= 刷新	F6= 建立	
F9= 检索	F10= 命令项	F23= 其余选项	F24= 其余键	
这是已划分子集列表。				

对于初学者，OS400 提供了非常便利的提示信息。在 PDM 环境，我们 F18, 然后在下边的全画面选项，选择 N，即可显示出如上图的相关帮助提示信息。

参照 PDM 处理库的显示界面上的帮组提示信息，我们可以对库进行建立，更改，删除，变更，检索等操作，请读者自己体会。

#### 2.2.4 处理对象

从 PDM 主菜单选 2，按执行键，出现规定处理对象的显示。键入对象所在的路径和对象名（支持模糊查询），执行，进入 PDM 处理对象的显示。



使用 PDM 处理对象					XXXXXXXX
库 . . . . .	TST010	定位至 . . . . .			
		定位至类型 . . . . .			
输入选项，按“ 执行” 键。 2= 更改      3= 复制      4= 删除      5= 显示      7= 重命名 8= 显示说明      9= 保存      10= 复原      11= 移动 ...					
Opt	对象	类型	属性	文本	
—	ORPGSRC	*FILE	PF-SRC	SRC FILE	
					底部
参数或命令 ===>					
F3= 退出	F4= 提示	F5= 刷新	F6= 建立		
F9= 检索	F10= 命令项	F23= 其余选项	F24= 其余键		

在此界面，我们同样可以根据画面的提示，对对象进行相应的处理。

### 2.2.5 处理成员

从 PDM 主菜单选 3，按执行键，出现规定处理成员的显示。键入成员的路径。则可列出对应参数下的成员列表。如下图：

使用 PDM 处理成员		XXXXXXXX
文件 . . . . .	<u>ODDSSRC</u>	
库 . . . . .	<u>TST010</u>	定位至 . . . . .
输入选项，按“ 执行” 键。 2= 编辑    3= 复制    4= 删除    5= 显示    6= 打印    7= 重命名 8= 显示说明    9= 保存    13= 更改文本    14= 编译    15= 建立模块 ...		
Opt	成员	类型
—	SBMAST	PF
—	SYMAST	PF
		文本
		设备管理表
		雇员管理表
参数或命令 ===>		底部
F3= 退出	F4= 提示	F5= 刷新
F9= 检索	F10= 命令项	F23= 其余选项
		F6= 建立
		F24= 其余键

在此界面，我们可以根据画面的提示，对成员进行相应的处理。

## 2.2.6 处理用户定义的选项

用处理用户定义的选项，可从 PDM 任何显示中调用用户自己的命令，这个选项能很容易的执行经常要做的操作，因此可以就写一个选项而不必写出整个命令。

下面给出 PDM 原有的用户定义的选项样版：

选项名	调用的命令	解 释
C	CALL &O/&N	允许在 PDM 处理成员显示中运行一个程序
CC	CHGCURLIB CURLIB(&L)	允许在 PDM 处理成员或对象显示中把此库做为当前库
CD	STRDFU OPTION(2)	允许生成一个 DFU 程序
CL	CHGCURLIB CURLIB(&N)	允许在用 PDM 处理库的显示中把此库做为当前库
CM	STRSDA OPTION(2) SRCFILE(&L/&F) ??SRCMBR()	允许用 SDA 生成成员（菜单）
CS	STRSDA OPTION(1) SRCFILE(&L/&F) ??SRCMBR()	允许用 SDA 生成成员（显示）
DM	DSPMSG	显示信息
EA	EDTOBJAUT OBJ(&L/&N) OBJTYPE(&T)	在 PDM 处理对象中编辑对象授权
GO	GO &L/&N	显示菜单



JL	DSPJOBLOG	显示作业日志
SL	SBMJOB ??CMD(SAVLIB LIB(&N))	在 PDM 处理库中用批处理保存库
SM	SBMJOB ??CMD(SAVOBJ OBJ(&F) LIB(&L) OBJTYPE(*FILE) FILEMBR((&F(&N))))	在 PDM 处理成员中用批处理保存成员
SO	SBMJOB ??CMD(SAVOBJ OBJ(&N) LIB(&L))	在 PDM 处理成员中用批处理保存对象
SP	WRKSPLF	允许处理假脱机文件
WS	WRKSBMJOB	允许处理批作业

#### 生成用户定义选项

在 PDM 主菜单选 9，按执行键，出现规定处理选项文件的显示，显示中的提示都缺省为活动的用户定义选项文件；执行；用 F6 键，出现生成用户定义选项的显示；

在 options 提示中给出表示命令的字符，第一个字符必须是字母，第二个字符可以是任何字母数字。在 command 提示中给出 CF 要调用的相应命令，按执行键。如果不记得命令的正确格式，用 F4 键，给出命令参数的提示。此时在显示底部有信息指出 CF 已生成好，新的选项保存在文件 QAU00PT 中。

#### 有效的用户定义选项替换变量

下表给出用在用户定义选项中有效的参数变量及每类列表返回的值：

参 数	意 义	说 明
&A	对象属性	如果处理对象, &A 由列表中的对象属性代替 如果处理库或成员, &A 由*NULL 代替
&B	列表类型	如果处理库列表, &B 由 X 代替 如果处理库清单, &B 由 L 代替 如果处理对象列表, &B 由 O 代替 如果处理成员列表, &B 由 M 代替
&C	选项	&C 由用户定义选项码代替
&D	成员修改日期	如果处理成员, &D 由成员最后修改日期代替 返回值用系统格式及分隔符, 否则&D 用*NULL 代替, 变量必须用' ' (即' &D' ), 这是因为它有特殊字符/, 它用做操作符。
&E	用批处理运行	如果处理成员, &E 用成员所在的文件名代替 否则, &F 用*NULL 代替
&F	文件名	如果处理成员, &F 由成员所在的文件代替, 对所有其它情况, 用*NULL 代替&F。
&G	作业描述库	&G 用修改缺省显示中的作业描述库代替
&H	作业描述库名	&H 用修改缺省显示中的作业描述值代替
&J	作业描述库	&J 用修改缺省显示中的作业描述值代替 格式为库/作业描述
&L	库名	如果处理库, &L 用 QSYS 代替 如果处理成员或对象, &L 用包括对象和成员的库名代替
&N	项目名	&N 由列表中进入选项边上的项目名代替
&O	对象库	如果处理库、对象或成员, &O 由修改缺省显示中的对象库代替
&P	用批方式编译	如果在修改缺省显示中, 在用批方式编译提示中给出 Y, 则&P 由*YES 代替。如果为 N, 由*NO 代替。
&R	替换对象	如果在修改缺省显示中, 在替换对象提示中给出 Y, 则&P 由*YES 代替。如果为 N, 由*NO 代替。
&S	无' *' 的项目类型	如果处理库, &S 由 LIB 代替 如果处理对象, &S 由不带*的对象类型代替 如果处理成员, &S 由成员类型代替
&T	有' *' 的项目类型	如果处理库, &T 由*LIB 代替 如果处理对象或成员, &T 由它们的类型代替
&U	用户定义选项文件	&U 由修改缺省显示的用户定义选项文件名代替



&V	用户定义选项文件库	&V 由修改缺省显示的用户定义选项文件库名代替
&W	用户定义选项文件成员	&W 由修改缺省显示的用户定义选项文件成员名代替
&X	项目说明	&X 由进入选项边上的项目说明代替(有' ' )

### 修改用户定义选项

可用处理用户定义选项显示中的选项 2 来修改它，当用这个选项时，不能在命令行输入任何内容。

### 复制用户定义选项

可把用户定义的选项用处理用户定义选项中的选项 3 复制到同一成员中，也可复制到同一库、文件的不同成员中或不同库及文件中。

注：如果复制到不同的成员、文件和库中，to file 提示在列表中的所有选项完成之后立即修改。在处理用户定义选项显示中的 copy 选项不是成组选项，即使用 F15 键（不保存结束）也做复制。如果复制到相同的库、文件和成员中，当在处理用户定义选项显示中用 F3、F12 或执行键结束时修改文件，用 F15 键取消对已有成员所做的修改。

### 删除用户定义选项

用 PDM，在 opt 列写 4，可以删除不再使用的选项，可删除一组选项也可删除某一个选项。

## 2.2.7 更改 PDM 缺省值

可用选择缺省显示来对 PDM 中某些操作选择缺省值，可用 PDM 菜单、F18 键或用下列任何显示来访问修改缺省的显示。如下图：

更改缺省值		
输入选择，按“ 执行” 键。		
对象库 . . . . .	<u>TST010</u>	名称, *CURLIB, *SRCLIB
置换对象 . . . . .	<u>N</u>	Y= 是, N= 否
以批处理方式编译 . . . . .	<u>N</u>	Y= 是, N= 否
以批处理方式运行 . . . . .	<u>N</u>	Y= 是, N= 否
保存对话缺省值 . . . . .	<u>Y</u>	Y= 是, N= 否
保存 / 复原选项 . . . . .	<u>1</u>	1= 单个, 2= 全部
作业说明 . . . . .	<u>QBATCH</u>	名称, *USRPRF, F4 显示列表
库 . . . . .	<u>*LIBL</u>	名称, *CURLIB, *LIBL
更改类型和文本 . . . . .	<u>N</u>	Y= 是, N= 否
选项文件 . . . . .	<u>QAU0OPT</u>	名称
库 . . . . .	<u>QGPL</u>	名称, *CURLIB, *LIBL
成员 . . . . .	<u>QAU0OPT</u>	名称
全屏幕方式 . . . . .	<u>N</u>	Y= 是, N= 否
尚有 ...		
F3= 退出      F4= 提示      F5= 刷新      F12= 取消		

下面介绍常用的选择设置。

## 对象库

可从修改 `object lib` 提示来规定把编译生成的对象放在由编译程序组成的特别库中。当编译成员时，生成一个对象，放这个对象的库是由在修改缺省值显示中的 `object lib` 提示中的值确定的。这个提示的缺省值为 `*srclib`。它指出编译的结果存在源库中，要把对象存在不同的库里，在这个提示中规定新库名，那么当用 PDM 编译成员时，生成的对象都存在于这个库中。由于每次进入，都会保留本用户上次变更的值。所以，要注意此值的正确性。

## 置换对象

如果编译结果生成的对象已经存在，可在修改缺省值显示的 `Replace obj` 提示中规定用新的对象来替代已存在的对象，新的对象放在 `obj lib` 提示中。

注：也可在生成命令中使用 `REPLACE` 参数。

为了替代已存在的编译成员，在 `Replace obj` 提示中写 Y，在调用生成命令前，让 PDM 删除已存在的对象，用新的替代，如果编译的结果对象已经存在，它会在调用生成命令前被替代。

## 全屏方式

所有 PDM 的显示都是在一屏中给出列表项、`opt` 和可用的功能键，可用修改缺省显示中的 `full screen mode` 提示改成有更多的列表项，但无 `opt` 和功能键的显示方式。在 PDM 菜单中用 F18 省值的显示；在 `full screen mode` 提示中写 Y，按执行键，出现 PDM 主菜单。如果以后不改此提示，PDM 的列表显示都用全屏方式。

## 修改执行键的缺省值

可在 `Exit list on ENTER` 提示中修改执行键的缺省值，它的缺省值为 Y，即允许用执行键结束列表屏。在 PDM 菜单中用 F18 键，用翻页键走到第二屏显示；在 `Exit lists on ENTER` 提示中写 N，按执行键，出现 PDM 菜单。修改完后，就不能用执行键从列表显示中出来，可用 F3 或 F12 键结束列表显示。

到此，程序设计开发管理工具 PDM 的基本应用就介绍完了。

## 2.3 源语句输入实用工具

### 2.3.1 启动 SEU

SEU 是处理源物理文件中源成员以及成员中记录的工具。启动 SEU 的方式有很多，可在用 PDM 处理成员的显示中选择 2 或 5 启动。可在程序员菜单中选 8，或先选 5，然后运行 `STRSEU`。可在 SDA 的处理显示记录的显示中选 2 启动。这里重点讲解用命令启动 SEU，及 SEU 的参数设置和常用操作。

在命令行键入 `STRSEU` 然后 F4，进入启动 SEU 显示。

Start Source Entry Utility (STRSEU)		
Type choices, press Enter.		
Source file . . . . .	*PRV _____	Name, *PRV
Library . . . . .	_____	Name, *LIBL, *CURLIB, *PRV
Source member . . . . .	*PRV _____	Name, *PRV, *SELECT
Source type . . . . .	*SAME _____	Name, *SAME, BAS, BASP...
Option . . . . .	*BLANK _____	*BLANK, ' ', 2, 5, 6
Text 'description' . . . . .	*BLANK _____	
<div style="text-align: right;">Bottom</div> <div> F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display  F24=More keys </div>		

为了方便讲解，F11，显示参数项目名如下图：

Start Source Entry Utility (STRSEU)		
Type choices, press Enter.		
Source file . . . . .	SRCFILE	*PRV _____
Library . . . . .		_____
Source member . . . . .	SRCMBR	*PRV _____
Source type . . . . .	TYPE	*SAME _____
Option . . . . .	OPTION	*BLANK _____
Text 'description' . . . . .	TEXT	*BLANK _____
<div style="text-align: right;">Bottom</div> <div> F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display  F24=More keys </div>		

### 2.3.2 相关参数设定

下面就主要输入项目作详细讲解。

#### SRCFILE

规定包括要编辑或生成的源成员的源物理文件的限定名。**\*LIBL**：用库列表来查找源文件。**\*CURLIB**：用作业使用的当前库查找源文件，如果没定义当前库，用 QGPL。库名：规定源文件所在的库名。源文件名：规定已有的源文件名。

#### SRCMBR

规定要编辑或生成的源成员名，这个参数的缺省值依赖于是否规定了 SRCFILE 参数。**\*PRV**：使用原来规定的源成员名。如果在 SRCFILE 中规定了**\*PRV**，那么这是缺省值。**\*SELECT**：SEU 给出成员列表，用户可以从这里选择一个来编辑、显示、打印或删除，如果 SRCFILE 中规定的不是**\*PRV**，则**\*SELECT** 是缺省值。源文件成员名：规定一个源成员名来生成或编辑。

#### TYPE

规定源成员的类型。**\*SAME**：对已有的成员，这个值指出类型没修改。对要生成的成员，它指出源成员的类型用源文件名相关的缺省值。详细内容请看 2.4.4。

类型：规定源成员的类型，可由 10 个字符组成，也可是下列 SEU 支持的类型之一。常用类型有：CL, CLP, CMD, DFU, DSPF, LF, MENU, MNU, MNUDDS, PF, QRY, RPG, RPGLE, SQLCLE, SQLRPG, SQLRPGLE, TXT。更多类型，请参照源成员类型表。

#### OPTION

规定对选择的成员完成的功能，缺省值依赖于是否规定了成员名。**\*BLANK**：如果没规定成员名，这是缺省值，它没有任何意义，只出现成员列表，从中用户可规定下列选项：2：编辑成员，如果规定了成员名，则 2 是缺省值；5：显示成员；6：打印成员。

#### TEXT

对这个成员进行指定性说明。这个说明存在成员的说明字段中。**\*BLANK**：这是新成员的缺省值，对已存在的成员，它不改变成员的说明字段。说明：规定不大于 50 个字符的说明，要放在引号内。

配置参数时，如果不填上边介绍的库名，等参数，系统会根据文件的不同而选择不同的类型。但建议最好键入成员类型，确保生成所要属性的成员。

#### 生成成员

在命令行，用命令提示，给出要生成的成员名及类型，OPTION 处键入 2，执行，出现编辑显示，这时，可以输入源语句了。

下面是 AS/400 支持的源成员类型：

说 明	类型
Auto report	RPT
BASIC	BAS
BASIC program	BASP
Binder Language	BND
C	C



C with embedded SQL	SQLC
C locale description	CLD
CICS C	CICSC
CICS maps	CICSMAP
CL	CL
CL program	CLP
COBOL	CBL
COBOL with embedded CICS/400* statements	CICSCBL
COBOL with embedded SQL	SQLCBL
COBOL with embedded SQL and CICS/400 statements	CICSSQLCBL
Command definition	CMD
DFU	DFU
Display	DSPF
FORTRAN/400* member	FTN
FORTRAN/400 member with embedded SQL	SQLFTN
ICF	ICFF
ILE C	C
ILE C with embedded SQL	SQLC
ILE CL	CLLE
ILE COBOL	CBLLE
ILE COBOL with embedded CICS/400 statements	CICSCBLLE
ILE COBOL with embedded SQL	CICSCBLLE
ILE RPG	RPGLE
ILE RPG with embedded SQL	SQLRPGLE
Logical file	LF
Menu	MNU
Menu (UIM)	MENU
Menu command source	MNUCMD
Menu DDS source	MNUDDS
Panel group source	PNLGRP
Pascal	PAS
Physical file	PF
PL/I	PLI

PL/I with embedded SQL	SQLPLI
Printer	PRTF
Query	QRY
REXX	REXX
RM/COBOL ** member	RMC
RPG	RPG
RPG with embedded SQL	SQLRPG
Sort	SRT
Spelling dictionary	SPADCT
Table	TBL
Text	TXT

### 显示成员

可用浏览显示来看一下成员且避免修改它，可用扫描或定位操作（象翻上页、下页），但不能做任何修改成员的操作（例如增加、删除、修改或移出记录）。

在 STRSEU 命令中的 option 参数规定为 5 或在用 SEU 处理成员的显示中选 5，出现选择的成员显示屏幕。

### 修改显示的环境

可在修改对话缺省显示中规定参数来修改显示环境（例如设置显示中翻页的记录数或强制大写输入），在浏览显示中用 F13 键可访问修改对话缺省显示。也可用 SEU 的 SET 命令来修改显示的环境。如下图

Change Session Defaults		
Type choices, press Enter.		
Amount to roll . . . . .	<u>C</u>	H=Half, F=Full C=Cursor, D=Data 1-999
Uppercase input only . . . . .	<u>Y</u>	Y=Yes, N=No
Tabs on . . . . .	<u>N</u>	Y=Yes, N=No
Increment of insert record . . . . .	<u>0.01</u>	0.01-999.99
Full screen mode . . . . .	<u>N</u>	Y=Yes, N=No
Screen size . . . . .	<u>2</u>	1=27x132, 2=24x80
Source type . . . . .	<u>PF</u>	
Syntax checking:		
When added/modi fied . . . . .	<u>Y</u>	Y=Yes, N=No
From sequence number . . . . .	_____	0000.00-9999.99
To sequence number . . . . .	_____	0000.00-9999.99
Set records to date . . . . .	<u>/ /</u>	YY/MM/DD or YYMMDD More...
F3=Exit F5=Refresh F12=Cancel F14=Find/Change options F15=Browse/Copy options		

### 浏览显示分屏和参照其他源程序

当显示一个成员时，可以分屏以便显示另一成员或假脱机文件。在日常程序开发，经常要用到参照另一本原成员，要分屏：用 F15 键，出现浏览选项显示，

Browse/Copy Options		
Type choices, press Enter.		
Selection . . . . .	<u>1</u>	1=Member 2=Spool file 3=Output queue
Copy all records . . . . .	<u>N</u>	Y=Yes, N=No
Browse/copy member . . . . .	<u>TEST01R</u>	Name, F4 for list
File . . . . .	<u>QRPGRSRC</u>	Name, F4 for list
Library . . . . .	<u>TST010</u>	Name, *CURLIB, *LIBL
Browse/copy spool file . . . . .	<u>TEST01R</u>	Name, F4 for list
Job . . . . .	<u>TEST01R</u>	Name
User . . . . .	<u>TST010</u>	Name, F4 for list
Job number . . . . .	<u>*LAST</u>	Number, *LAST
Spool number . . . . .	<u>*LAST</u>	Number, *LAST, *ONLY
Display output queue . . . . .	<u>QPRINT</u>	Name, *ALL
Library . . . . .	<u>*LIBL</u>	Name, *CURLIB, *LIBL
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=Change session defaults F14=Find/Change options		



填写好要查看的成员名，文件，库，执行。即可出现一下显示画面：

Columns . . . :	1 80	Edi t	TST010/QRPGSRC
SEU==>			TEST01R
FMT *	..... * 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8		
	***** Beginning of data *****		
0001.00	F*****		081013
0002.00	FTEST01D CF E	WORKSTN	081013
0003.00	FSYMAST IF E	DISK	081013
0004.00	E*****		081013
0005.00	E	#CMD 1 1 78	COMMAND KEY 081013
0006.00	E	#EER 1 2 78	COMMAND KEY 081013
0007.00	I*****		081013
0008.00	I SDS		081013
SEU==>			TST010/QRPGSRC
			TEST01R
	***** Beginning of data *****		
0001.00	F*****		081013
0002.00	FTEST01D CF E	WORKSTN	081013
0003.00	FSYMAST IF E	DISK	081013
0004.00	E*****		081013
0005.00	E	#CMD 1 1 78	COMMAND KEY 081013
0006.00	E	#EER 1 2 78	COMMAND KEY 081013
0007.00	I*****		081013
F3=Exit	F4=Prompt	F5=Refresh	F9=Retrieve
F16=Repeat find	F17=Repeat change	F11=Toggle	F12=Cancel
		F24=More keys	

### 用日期查找记录

要找到某个日期时最后修改的记录：用 F14 键，出现查找显示；在 Search on date 提示中给出日期；在 Compare 提示中给出 1、2 或 3；用 F16 键，即找规定日期的记录，再用 F16 键（连续查找），继续查找其它满足条件的记录。

Find/Change Options		
Type choices, press Enter.		
Find . . . . .		
Change . . . . .		
From column number . . . . .	<u>1</u>	1-80
To column number . . . . .	<u>80</u>	1-80 or blank
Occurrences to process . . . . .	<u>1</u>	1=Next, 2=All, 3=Previous
		4=First, 5=Last
Records to search . . . . .	<u>1</u>	1=All, 2=Excluded
		3=Non-excluded
Kind of match . . . . .	<u>1</u>	1=Same case
		2=Ignore case
Allow data shift . . . . .	<u>N</u>	Y=Yes, N=No
Search for date . . . . .	<u>08/10/13</u>	YY/MM/DD or YYMMDD
Compare . . . . .	<u>-</u>	1=Less than
		2=Equal to
		3=Greater than
F3=Exit	F5=Refresh	F12=Cancel
F15=Browse/Copy services	F16=Find	F17=Change session defaults
		F18=DBCS conversion

同理，还有按找串查找。用日期查找和找串是互相排斥的。Compare 提示决定当按 F16 键时，



做哪类查找。如果此提示空白，SEU 就查找串。如果不空，则查找日期。

### 查找及修改串

在查找显示或查找/修改显示中的 Find 提示中给出要查找的串；用 F16 键，即找串。如下图

Columns . . . : 1 80	Edi t	TST010/QRPGSRC
SEU==>EXFMT		TEST01R
FMT *	*. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8	
*****Beginni ngofdata*****		
0001.00	F*****	081013
0002.00	FTEST01D CF E	081013
0003.00	FSYMAST IF E K DISK	081013
0004.00	E*****	081013
0005.00	E #CMD 1 1 78	081013
0006.00	E #EER 1 2 78	081013
0007.00	I*****	081013
0008.00	I SDS	081013
0009.00	I 1 10 W#PGID	081013
0010.00	I 244 253 W#WSID	081013
0011.00	I 254 263 W#USID	081013
0012.00	C*****	081013
0013.00	C*MIAN	081013
0014.00	C*****	081013
0015.00	C DO *HIVAL	081013
0016.00	C #CTL CASEQ' ' @INZ	081013
0017.00	C #CTL CASEQ' FMT01' @FMT01	081013
0018.00	C #CTL CASEQ' END' @END	081013
0019.00	C END	081013
F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle		
F16=Repeat find F17=Repeat change F24=More keys		

F16 后找到后，光标停留在该字符串：

Columns . . . : 1 80	Edi t	TST010/QRPGSRC
SEU==>		TEST01R
FMT C	.....CLON01N02N03Factor1+++OpdcFactor2+++Resul tLenDHHI LoEqComments+++++.....	
0039.00	C WRIT FMT99	081013
0040.00	C EXFMT FMT01	081013
0041.00	C*	081013
0042.00	C SELEC	081013
0043.00	C *INKC WHEQ *ON	081013
0044.00	C *INKL OREQ *ON	081013
0045.00	C MOVE' END' #CTL P	081013
0046.00	C MOVE' *BLANK W#LOP1 P	081013
0047.00	C OTHER	081013
0048.00	C EXSR @CHK1	081013
0049.00	C ENDSL	081013
0050.00	C ENDDO	081013
0051.00	C*	081013
0052.00	C ENDSR	081013
0053.00	C*****	081013
0054.00	C*@INZ1	081013
0055.00	C*****	081013
0056.00	C @INZ1 BEGSR	081013
0057.00	C*	081013
0058.00	C EXSR @RST	081013
F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle		
F16=Repeat find F17=Repeat change F24=More keys		
String EXFMT found.		

### 命令行常用命令

SEU 支持有效的命令为：

命 令	功 能
FIND 或 F	在成员中找一个字符串

HIDE 或 H	从显示中隐藏包括特定字符串的记录
CHANGE 或 C	在成员中找一个字符串，且把它改为另一个串
SET 或 S MATCH	在查找操作中找匹配的串
SET 或 S SHIFT	在修改操作中，当代替一个串时要左右移数据
SET 或 S CAPS	把输入数据的类型设为大写或大小写都可
SET 或 S TABS	使用预先定义的标号位置
SET 或 S ROLL	设置显示上、下、左、右移动的总数
SET 或 S EXPERT	设显示为全屏方式
TOP 或 T	把工作屏定位在数据的第一页
BOTTOM 或 BOT 或 B	把工作屏定位在数据的最后一页
CANCEL 或 CAN	取消对话，且结束或取消分屏底部的显示对话，在顶部对话的 cancel 结束两个对话
SAVE	保存对成员所做的修改，且继续编辑
FILE	保存对成员所做的修改，结束 SEU

#### 移动，拷贝，删除记录

After(A)，Before(B)和 Overlay(O)命令是对象位置定位命令。可用它们做复制(C)，连续复制(CR)，移动(M)或块(MM、CC)命令的对象位置。注意：MM、CC、CCR 命令成对出现，一对命令之间的一块记录为处理记录。

#### 插入空记录

可在成员中加一些空行以增加新记录。可用下列行命令加一个或多个空行。记录的顺序号处键入 I：在这个记录下加一个空行，每次在这行写完数据，按执行键时，SEU 加另一空行。键入 In：在这个记录下加 n 个空行，当在最后一行写完数据，按执行键时，SEU 加另一空行。

在记录顺序号处可以输入很多命令，可以把光标放置到顺序号处，按 F1 查询命令列表和帮助信息。

下表列出 SEU 行命令，所有行命令对全屏编辑和分屏编辑都有效。表中指出哪些行命令对浏览有效。

命 令	浏览显示	行命令									
绝对位置	Y	N	. n	n. n							
后，前	N	A	An	B	Bn						
列	Y	COL									
		S									
复制	Y/N(1)	C	Cn	CC							
连续复制	Y/N(1)	CR	CRn	CCR							
删除	Y/N(2)	D	Dn	DD							
取消	Y	X	Xn	XX							

格式	Y	F	F?	Fxx							
插入	N	I	In								
插入格式	N	IF	IFn	IF?	IF? n	IFx x	IFxx n				
插入提示	N	IP	IP?	IPx x							
行打印	Y	LP	LPn	LLP							
移动	N	M	Mn	MM							
重叠	N	O	On	OO							
提示	N	P	P?	Pxx							
相对位置	Y	+	+n	-	-n						
重复	N	RP	RPn	RPP	RPP n						
左移	N	L	Ln	LL	LLn						
右移	N	R	Rn	RR	RRn						
有截断的左移	N	LT	LTn	LLT	LLT n						
有截断的右移	N	RT	RTn	RRT	RRT n						
出示	Y	SF	SFn	SL	SLn						
轮廓	N	S	IS	ISn							
标号	N	TAB S									
窗口	Y	W	Wn								

注：1. 仅对在分屏编辑/显示中在编辑部分有对象命令时有效；2. 仅对特殊记录有效（例如格式行）。

## 第三章 数据库

AS/400 DB2 通用数据库(UDB/400)是集成在 OS400 中的数据库管理系统(DBMS)。DB2 UDB for AS/400 通用数据库，不但支持绝大多数开放数据库的标准，而且可以使关系数据库能以一种单一连贯的数据库结构形式来存储、管理、索引和控制各种形式的信息。是目前应用最广泛的数据库之一。本章主要从程序员角度，通过实例，讲述如何建立数据库文件。

### 3.1 数据库文件基本概念

#### 3.1.1 数据库文件定义方法

使用以下其中的一种就可以执行数据库数据定义语言(Data Definition Language, DDL)操作：  
DDS (数据描述规范, Data Description Specification) :

DDS 是 AS/400 专有的本机接口，最初是在 SQL 普及之前，在数据库中使用的。现在仍大量使用于数据库操作。

SQL :

使用 SQL(例如, SQL 存储过程、触发器和函数)进行应用程序开发需要用户购买 Query Manager 和 DB2 SQL Development Kit。有了 V5R2, 则不再需要购买这个工具箱, 因为该功能已经添加到 iSeries, 作为其基本产品的一部分。要获得所有其它 SQL 支持, 比如预编译器、ISQL 接口和 Query Manager, 这一工具箱仍需付费。

不管使用了什么方法, 任何编程语言或数据库接口都应该能互换地使用这些方法来访问 DB2 UDB for iSeries 的数据。因此, 例如 SQL 语句就可以引用用 DDS 创建的对象。下面的表比较了 DDS 和 SQL 的概念和术语。

DB2 UDB for iSeries 的 DDS vs SQL 概念和术语比较

DDS	SQL
与 iSeries 一起提供, 免费	与 iSeries 一起提供, 免费
能很好地处理 DDL, 但不能处理 DML	能处理 DDL 和 DML
库	模式 (也叫做 **集合, Collection)
物理文件	表
逻辑文件	视图
按照键索引的逻辑文件	索引
记录	行
字段	列
能用于创建屏幕和报表	基于 SQL 输出的基本“报表”
用于引用库中文件的表示法是: <库>/<文件名> 这称为 *SYS 命名约定。	用于引用作为模式一部分的表的表示法是: <模式>.<表名> 这称为 *SQL 命名约定。



### OS/400 交互式数据定义实用程序(DDU):

用 DDU 可以描述物理文件，DDU 是一种菜单驱动、交互式描述数据的方式。你可能很熟悉系统 36 上用 DDU 描述数据的方法，另外，DDU 允许描述多格式物理文件，以便用于查询、客户访问/400 和 DFU。

当使用 DDU 描述文件时，文件定义就成为 OS/400 数据字典的一部分。用 DDU 去描述一个文件，然后用 DDU 生成，所生成的文件就是外部说明文件。还可以把存在外部描述文件中的文件描述转入到数据字典中，系统总能保证数据字典中的定义和外部说明文件中的描述是一致的。

#### 3.1.2 物理文件:

类型为 \*FILE，属性为 PF-DTA 的对象：在 SQL 中称做表 (TABLE)，含有系统实际存储的数据，每个物理文件只有一个固定长度的记录格式。物理文件可以指定键值来规定其顺序存取路径，而不按记录写入时的物理顺序操作。

#### 3.1.3 逻辑文件:

类型为 \*FILE，属性为 LF 的对象：在 SQL 中称做表 (VIEW)，它不含有实际数据，依附于物理文件，可以描述一个或多个物理文件的记录 (JOIN 逻辑文件)。

逻辑文件可以改变物理文件中定义的字段属性 (如字段名和字段排列顺序)、提供记录的逻辑顺序、只选择物理文件中部分字段进行显示或修改、对字段值进行选择记录进行操作、从物理文件的字段基础上演绎新字段、联合多个物理文件。

在本章中我们着重介绍用 DDS 来创建外部描述数据库文件。关于 SQL 创建数据库文件，请参照第九章 SQL 的应用部分。

### 3.2 建立物理文件

使用源语句录入工具 SEU，通过 DDS 描述方式，创建 PF 类型的成员，然后编译生成 \*FILE 类型，PF-DTA 属性的物理文件，是创建外部描述数据库的常用方法。在本节，我们先实例一个‘雇员基本信息表’的物理文件，然后实际创建此表，以此详细讲述物理文件创建过程，和其中的注意事项。

#### 3.2.1 典型物理文件程序范例

下图，是员工基本信息表的源程序。

Columns	1	80	Edit	TST010/QDDSSRC
SEU==>				
FMT PF	.....A.....	T.Name+++++RLen++TDPB.....	Functions+++++	SYMAST
***** Beginning of data *****				
0001.00	A		UNIQUE	081013
0002.00	A	R SYMASTR		081013
0003.00	A	SYSYCD 10A	COLHDG(' 雇员 I D ')	081013
0004.00	A	SYSYNM 200	COLHDG(' 雇员名 ')	081013
0005.00	A	SYSYBT 40	COLHDG(' 性别 ')	081013
0006.00	A	SYSNDT 8S 0	COLHDG(' 出生年月日 ')	081013
0007.00	A	SYSNTY 3S 0	COLHDG(' 身高 ')	081013
0008.00	A	SYTAYX 3S 0	COLHDG(' 体重 ')	081013
0009.00	A	SYBMCD 3A	COLHDG(' 部门 ID ')	081013
0010.00	A	SYSYKY 3A	COLHDG(' 职位 ')	081013
0011.00	A	SYNYDT 8S 0	COLHDG(' 入社日期 ')	081013
0012.00	A*			081013
0013.00	A	K SYSYCD		081013
***** Endofdata *****				
F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle F16=Repeat find F17=Repeat change F24=More keys				

编译后的物理文件(图中 SYMAST 的表)

使用 PDM 处理对象				XXXXXXXX
库 . . . . .	TST010	定位至 . . . . .		
		定位至类型 . . . . .		
输入选项, 按“ 执行” 键。				
2= 更改	3= 复制	4= 删除	5= 显示	7= 重命名
8= 显示说明		9= 保存	10= 复原	11= 移动 ...
Opt	对象	类型	属性	文本
—	QDDSSRC	*FILE	PF-SRC	
—	QRPGRSRC	*FILE	PF-SRC	SRC FILE
—	SYMAST	*FILE	PF-DTA	雇员基本信息表
				底部
参数或命令				
==>				
F3= 退出	F4= 提示	F5= 刷新	F6= 建立	
F9= 检索	F10= 命令项	F23= 其余选项	F24= 其余键	

### 3.2.2 物理文件的代码范例

一般情况下，物理文件对应的源程序，都建立在专门放置数据库描述成员的源文件中。为了管理，通常会把这个源文件起名为 QDDSSRC。我们也把员工信息表(SYMAST)的描述文件，建立在 QDDSSRC 下。

首先，我们要启动源语句输入实用程序（SEU）。在命令行键入 STRSEU，然后按 F4，配置参数，

如下图：

Start Source Entry Utility (STRSEU)

Type choices, press Enter.

Source file . . . . .	> QDDSSRC	Name, *PRV
Library . . . . .	> TST010	Name, *LIBL, *CURLIB, *PRV
Source member . . . . .	> SYMAST	Name, *PRV, *SELECT
Source type . . . . .	*SAME	Name, *SAME, BAS, BASP...
Option . . . . .	> 2	*BLANK, ' ', 2, 5, 6
Text 'description' . . . . .	' 雇员基本信息表 '	

Bottom

F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display  
F24=More keys

### 参数说明

源文件：建立位置说明之一。我们按默认规则，放在 QDDSSRC 文件下。

库：建立位置说明之一。我们放在 TST010 库下的 QDDSSRC 下。

源成员：为此文件的名称。

类型：我们建的是物理文件，所以，此处 PF

处理选择（OPT）：键入 2（编辑）

文件注释：键入文件说明信息，或文件名

配置好后，回车进入 SEU 编辑界面



```
Columns . . . . : 1 80 Edit TST010/QDDSSRC
SEU==>
SYMAST
FMT PF . . . . A . . . . . T. Name+++++RLen++TDpB. . . . Functions+++++
*****Beginni ngofdata*****
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
*****Endof data*****
F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle
F16=Repeat find F17=Repeat change F24=More keys
Member SYMAST added to file TST010/QDDSSRC. +
```

因为，生成的类型中，我们键入的是 PF 类型，所以，进入编辑界面后，画面显示提示信息是 PF 的格式。

### 3.2.3 DDS 规则讲解

```
Columns . . . . : 1 80 Edit TST010/QDDSSRC
SEU==>
SYMAST
FMT PF . . . . A . . . . . T. Name+++++RLen++TDpB. . . . Functions+++++
*****Beginni ngof*****data
0001.00 A UNIQUE 081013
0002.00 A R SYMASTR 081013
0003.00 A SYSYCD 10A COLHDG(' 雇员 I D ') 081013
0004.00 A SYSYNT 200 COLHDG(' 雇员名 ') 081013
0005.00 A SYSYBT 40 COLHDG(' 性别 ') 081013
0006.00 A SYSNDT 8S 0 COLHDG(' 出生年月日 ') 081013
0007.00 A SYSNTY 3S 0 COLHDG(' 身高 ') 081013
0008.00 A SYTAYX 3S 0 COLHDG(' 体重 ') 081013
0009.00 A SYBMCD 3A COLHDG(' 部门 ID') 081013
0010.00 A SYSYKY 3A COLHDG(' 职位 ') 081013
0011.00 A SYNYDT 8S 0 COLHDG(' 入社日期 ') 081013
0013.00 A K SYSYCD 081013
Prompt type . . . PF Sequence number . . . 0013.00

Name Data Deci mal
Type Type Name Ref Length Type Positions
Use
R SYMASTR
Functions

F3=Exit F4=Prompt F5=Refresh F11=Previous record
F12=Cancel F23=Select prompt F24=More keys
```

根据上面的代码图，我们按从左到右的顺序讲解。

A 列：

在提示行 FMT PF 标记的，‘A’ 列，在一下任何一行记录，都要写 A，标记着这一行为有效记录。在‘A’ 后加\*，表明此行为注释代码。如 12 行即为注释行。

**T 列：**

类型项，此列有两种类型，R 说明定义了一个记录格式，如第 2 行的 T 列代码，紧跟其后边的是记录名，在描述字段时，此列不入力。如 3-11 行的代码。K 说明定义了一个键值。如 13 行代码标记。此表内将按照此字段排序。默认为升序。

**NEME 列：**

名前项：此列分三种情况，一种为记录格式名。如第 2 行代码，说明的是此表的记录格式名为 SYMASTR。第二种为字段名。如 3-11 行的代码，此处的含义为字段的定义名。字段名不可以重复。最大长度为 10 位。

**R 项：**

参照项：参照定义（R 表示只有当 T 列为空，NEME 列非空时有效，表示当前字段是一个参照字段，BLANK=当前字段非参照字段）

**LEN 项：**

长度项：字段长度。

**数据类型 T 项：**

常用的有效的数据类型有 A（字符）型，P（压缩十进制）型，S（区位十进制）型和 0（全角半角混用）型。下表给出数据文件有效的数据类型：

有效的数据类型表

字符	含 意
A	字符型
P	压缩十进制
S	区位十进制
B	二进制
F	浮点型
H	十六进制
L	日期型
T	时间型
Z	时间标记

注意：1. 0 型必须要偶数位。2. AS/400 系统做算术操作时，使用压缩十进制比区位十进制更有效。3. 某些高级语言不支持浮点数据。

如果没有指定数据类型，则小数位项通常用于确定数据类型。如果小数位为空，数据类型为字符型(A)；如果该位是 0 到 31 的数，则数据类型为压缩十进制(P)。

如果一个高级语言程序使用了压缩十进制或区位十进制字段，字段长度必须限定在此高级语言所允许的长度，这个长度并不是存储区中字段的长度，而是在存储器外部指定的字符或数的位数，例如：一个 5 位数的压缩十进制字段在 DDS 中指定的长度为 5 位，但在存储区中只占用了 3 个字节。

**DP 项：**

小数点位数项：小数点位置定义，当数据类型 T 列定义为非数字型字段时，本栏应为空

#### FUNCTIONS 项：

功能定义，用于定义各种关键字

关键字的使用大大丰富了 DDS 的文件定义。分为

##### 1. 文件级，位于记录名之前，主要包括

REF：指定被参考文件

UNIQUE：键值唯一（如本实例代码的第 1 行），定义后，数据不能有重复的键值数据，否则报错。

FIFO, FCFO, LIFO：相同键值记录的排列规则，有 UNIQUE 时，就不能用。

##### 2. 记录级，位于记录名和第一个字段名之间，主要有：

FORMAT：共享其他文件的记录格式。

TEXT：记录格式说明。

##### 3. 字段级，关键字位于一个字段名和其下一个字段名或第一个关键字段名之间，包括：

COLHDG：为数据库文件中字段显示和打印用。

REFFLD：参考一个已定义的字段，允许字段长度和参考的字段不同，可用+N 或-N 来改变字段长度。

VARLEN：可变长度字段定义。

关键字段级，关键字位于第一关键字段名和其下一个关键字段名或成员尾之间，主要包括：

DESCEND：按降序排列，可用于字符或数值型键字字段。

ABSVAL：按绝对值顺序排序。

下面，我们给出数据文件有效的键字和简要的说明，读者可以在 FUNCTIONS 项上 F1 查看对应的详细说明。

键字	简要功能说明	键字	简要功能说明
ABSVAL	绝对值	EDTWRD	编辑字
ALIAS	替换名	FCFO	先修改先出
ALL	全部（仅限逻辑文件）	FIFO	先进先出
ALTSEQ	交替分配顺序	FLTPCN	浮点精度
ALWNULL	允许空值（仅限物理文件）	FORMAT	格式
CCSID	编码字符集标识（仅限物理文件）	LIFO	后进先出
CHECK	检查	NOALTSEQ	无交替顺序
CHKMSGID	检查信息标识	RANGE	范围
CMP	比较	REF	引用（仅限物理文件）
COLHDG	栏对象题	REFFLD	被引用字段（仅限物理文件）
COMP	比较	REFSHIFT	引用换档
CONCAT	连接（仅限逻辑文件）	RENAME	重命名（仅限逻辑文件）
DATFMT	日期格式	SIGNED	符号
DATSEP	日期分隔符	SST	子串（仅限逻辑文件）

DESCEND	降序	TEXT	正文
DFT	缺省值（仅限物理文件）	TIMFMT	时间格式
DIGIT	数字	TIMSEP	时间分隔符
DYNSLT	动态选择（仅限逻辑文件）	TRNTBL	转换表（仅限逻辑文件）
EDTCDE	编辑码	UNIQUE	唯一
EDTWRD	编辑字	UNSIGNED	无符号
FCFO	先修改先出	VALUES	值
FIFO	先进先出	VARLEN	变长字段
FLTPCN	浮点精度	ZONE	零

在定义表过程中，可以在上边的命令行，键入 SAVE，保存当前。也可键入 FILE 命令，保存当前代码，退出。

### 3.2.4 生成物理文件

定义好源物理文件后，可以建立物理文件，来对数据进行承载。用 CRTPF 命令创建物理文件。在 PDM 显示界面命令行，键入 CRTPF（也可在源物理文件前 OPT 键入 14），F4, 配置参数如下：

Create Physical File (CRTPF)

Type choices, press Enter.

File . . . . .	>	<u>SYMAST</u>	Name	
Library . . . . .	>	<u>QGPL</u>	Name, *	CURLIB
Source file . . . . .	>	<u>QDDSSRC</u>	Name	
Library . . . . .	>	<u>TST010</u>	Name, *	LIBL, *CURLIB
Source member . . . . .	>	<u>SYMAST</u>	Name, *	FILE
Record length, if no DDS . . . .			Number	
Generation severity level . . . .		<u>20</u>	0-30	
Flagging severity level . . . . .		<u>0</u>	0-30	
File type . . . . .		<u>*DATA</u>	*DATA, *	SRC
Member, if desired . . . . .		<u>*FILE</u>	Name, *	FILE, *NONE
User specified DBCS data . . . .		<u>*NO</u>	*NO, *	YES
Text 'description' . . . . .		<u>' 雇员基本信息表 '</u>		

---

More...

F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display  
F24=More keys

F10 键增加参数设定，可以 PAGUP, PAGEDOWN 上下翻页。

Create Physical File (CRTPF)

Type choices, press Enter.

Additional Parameters

Source listing options . . . . .		*SRC, *NOSRC, *SOURCE...
+ for more values		
System . . . . .	*LCL	*LCL, *RMT, *FILETYPE
Expiration date for member . . .	*NONE	Date, *NONE
Maximum members . . . . .	1	Number, *NOMAX
Access path size . . . . .	*MAX1TB	*MAX1TB, *MAX4GB
Access path maintenance . . . .	*IMMED	*IMMED, *DLY, *REBLD
Access path recovery . . . . .		*NO, *AFTIPL, *IPL
Force keyed access path . . . . .	*NO	*NO, *YES
Member size:		
Initial number of records . . .	*NOMAX	1-2147483646, *NOMAX
Increment number of records . .	1000	Number
Maximum increments . . . . .	3	Number
More...		
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display		
F24=More keys		

### 相关参数详解

文件名：编译后生成的数据表名称。

库名：指定数据表生成库位置

源程序文件名：源代码所在文件名称。

源文件库：源代码所在库

源程序：源代码名

成员最大数(参数列表第 2 页的 MAXMBRS 项)：如果建立物理文件时，要求有多成员，则要修改 MAXMBRS 参数值，可以指定为无限定 (\*NOMAX)

数据库容量（位于参数列表第 2 页的 SIZE 项）：

初期记录数指数据库首次可以存储数量，并分配相应存储空间，默认为 10000 条。当大于 10000 时，系统会提示确认信息。被确认后，会增加下边的增分记录数的存储空间，默认为 1000 条。直到增加到最多增加次数，默认为 3 次。此处涉及到数据库的高级应用。一般根据此表的功能，业务的需求而设置不同。由于此项有提示确认信息，可能会使程序执行终止或报错。因此，为了安全和程序的健壮性，当根据业务不能确认此表的数据数量，又没有特殊处理时，常常设置为 \*NOMAX。

其他参数，根据实际要求处理，一般为默认即可。

### 编辑错误处理

配置完后，回车执行。如果有错误，会有信息显示，DDS 错误。如果没有错误，会提示对象被作成。发生错误后，通常要查看错误信息。在命令行处键入 WRKJOB OPTION(\*SPLF) 命令，然后执行，会出现本用户本工作站的打印消息队列。如图：



Work with Job Spooled Files							
Job:	DMK079	User:	TST010	Number:	023641		
Type options, press Enter.							
1=Send 2=Change 3=Hold 4=Delete 5=Display 6=Release 7=Messages							
8=Attributes 9=Work with printing status							
Opt	File	Device or Queue	User Data	Status	Total Pages	Current Page	Copies
—	SYMAST	OPRINT		RDY	3		1
—	SYMAST	OPRINT		RDY	3		1
—	SYMAST	OPRINT		RDY	3		1
Bottom							
Parameters for options 1, 2, 3 or command							
===>							
F3=Exit F10=View 3 F11=View 2 F12=Cancel F22=Printers F24=More keys							

找到该表的信息后（如果编译了多次，最下边的是最后一次），在前边键入 5，执行

Display Spooled File					
File . . . . .	SYMAST	Page/Line	1/1		
Control . . . . .		Columns	1 - 127		
Find . . . . .					
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...0...+...1...+...2...+...					
5722SS1 V5R2M0 020719	Data Description	TST010/SYMAST	8/10/13 11:56:30	Page	
File name . . . . .	SYMAST				
Library name . . . . .	TST010				
File attribute . . . . .	Physical				
Source file containing DDS . . . . .	QDDSSRC				
Library name . . . . .	TST010				
Source member containing DDS . . . . .	SYMAST				
Source member last changed . . . . .	08/10/13 11:38:08				
Source listing options . . . . .	*SOURCE *LIST *NOSECLVL *NOEVENTF				
DDS generation severity level . . . . .	20				
DDS flagging severity level . . . . .	00				
File type . . . . .	*DATA				
Authority . . . . .	*LIBCRTAUT				
Replace file . . . . .	*NO				
Text . . . . .	雇员管理表				
Compiler . . . . .	IBM AS/400 Data Description Processor				
Data Description Source					
SEQNBR *...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8	Date				
100 A	UNIQUE	08/10/13			
More...					
F3=Exit F12=Cancel F19=Left F20=Right F24=More keys					

这里边详细记载着此表的编译错误信息。在控制命令行键入 B，到文件的最后部分

Display Spooled File									
File	SYMAST							Page/Line	1/35
Control	B							Col umns	1 - 127
Find									
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...0...+...1...+...2...+...									
1200	A	K SYSYCD					08/10/13		
***** END OF SOURCE *****									
5722SS1 V5R2MO	020719	Data Description					TST010/SYMAST	8/10/13 12:01:25	Page
Expanded Source									
SEQNBR	*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8 length	Field	Buffer position						
100	UNIQUE					Out	In		
***** END OF EXPANDED SOURCE *****									
5722SS1 V5R2MO	020719	Data Description					TST010/SYMAST	8/10/13 12:01:25	Page
Messages									
ID	Severity	Number							
* CPD7420	30	1	Message . . . . : Positional values found before first record.						
5722SS1 V5R2MO	020719	Data Description					TST010/SYMAST	8/10/13 12:01:25	Page
Message Summary									
Total	Informational	Warning	Error	Severe					
	(0-9)	(10-19)	(20-29)	(30-99)					
1	0	0	0	1					
* CPF7302	40	Message . . . . : File SYMAST not created in library TST010.							
***** END OF COMPI LATION *****									
Bottom									
F3=Exit	F12=Cancel	F19=Left	F20=Right	F24=More keys					

信息合计部分，记录了错误的个数。信息部分，详细描述了错误的原因和等级。由于高等级错误可能会衍生低等级错误。因此，排除错误，通常按由高到低的顺序排除。

### 3.3 建立逻辑文件

#### 3.3.1 逻辑文件种类

逻辑文件的结构可以很简单，也可很复杂。主要有四类：

1. 简单逻辑文件，把单个物理文件或表映射到逻辑记录定义
2. 多重格式逻辑文件，允许对几个物理文件存取，每个物理文件都有自己的记录格式定义。这种逻辑文件只能通过 DDS 创建，不能用 SQL 创建。
3. 联接逻辑文件，从多个物理文件或表、逻辑文件或视图中组合字段，给出单个记录定义
4. SQL 视图类似于联接逻辑文件，和联接逻辑文件的区别主要是实现方法不一样，联接文件对每次联接进行维护或共享存取路径，而 SQL 视图在运行时通过查询定义模板找到需要的存取路径。用 SQL 建立逻辑文件的方法，请参照第九章 SQL 的应用。

与物理文件一样，使用源语句录入工具 SEU，通过 DDS 描述方式，创建 LF 类型的成员，然后编译生成\*FILE 类型，LF 属性的逻辑文件，是创建逻辑文件的常用方法。由于单个物理文件表映射到逻辑文件，是最常用的逻辑文件形式。因此本节，将以上节的物理文件为基础，实例一个‘雇员信息表’的逻辑文件，然后实际创建此表，以此详细讲述逻辑文件创建过程，和其中的注意事项。

#### 3.3.2 简单逻辑文件程序范例

逻辑文件实例:要调查公司内某部门下女员工的信息，员工号由小到大，岗位级别由大到小，建立逻辑文件。如下图，是按照此条件编写的逻辑文件代码。

Columns . . . : 1 80	Edit	TST010/QDDSSRC
SEU==>		SYMASTL1
FMT LF . . . . A . . . . . T.Name+++++. Len++TdPb. . . . . Functions+++++		
*****Beginnigof data *****		
0001.00	A	R SYMASTR
0002.00	A	K SYBMCD
0003.00	A	K SYSYKY
0004.00	A	S SYSYBT
		COMP(EQ '女')
*****Endof data *****		
F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle		
F16=Repeat find F17=Repeat change F24=More keys		

生成后的逻辑文件为

使用 PDM 处理对象		XXXXXXX
库 . . . . .	TST010	定位至 . . . . .
		定位至类型 . . . . .
输入选项，按“ 执行” 键。		
2= 更改	3= 复制	4= 删除
5= 显示	7= 重命名	
8= 显示说明	9= 保存	10= 复原
11= 移动 ...		
Opt	对象	类型
—	OCLPSRC	*FILE
—	QDDSSRC	*FILE
—	ORPGSRC	*FILE
—	SYMASTL1	*FILE
		属性
		PF-SRC
		PF-SRC
		PF-SRC
		LF
		文本
		CLP
		SRC FILE
		雇员基本信息表逻辑文件 1
底部		
参数或命令		
===>		
F3= 退出	F4= 提示	F5= 刷新
F9= 检索	F10= 命令项	F23= 其余选项
		F6= 建立
		F24= 其余键

物理文件中的数据为:

雇员 ID	雇员名	性别	生年日期	身高	体重	部門 ID	職位	入社日期
0000000001	梁毅超	男	19720102	181	80	001	008	19990612
0000000002	司徒馬	男	19740105	180	80	001	006	20000412
0000000003	恚夕禾	女	19740702	163	52	001	006	20000412
0000000004	陳凡	女	19740802	171	54	001	006	20000215
0000000005	稼萌	女	19760612	160	50	001	004	20004372

0000000006	倪佳	女	19780102	170	70	001	003	20010220
0000000007	于卉	女	19780102	165	50	001	003	20004372
0000000008	張一風	女	19780311	159	50	001	003	20024372
0000000009	范可	男	19780402	180	80	001	003	20020403
0000000010	唐煜	女	19780501	160	50	001	003	20020606
0000000011	唐連	男	19810104	170	62	001	002	20030307
0000000012	高逸菲	女	19810713	170	53	001	002	20030307
0000000013	野賞	男	19821202	168	63	001	002	20040607
0000000014	李想	男	19821202	179	80	001	002	20040607
0000000015	徐陟	女	19830102	160	60	001	001	20050606
0000000016	周超	女	19830704	170	56	001	001	20050606

按照逻辑文件，数据的顺序和结果为

雇员 ID	雇员名	性別	生年日期	身高	体重	部門 ID	職位	入社日期
0000000003	倪夕禾	女	19740702	163	52	001	006	20000412
0000000004	陳凡	女	19740802	171	54	001	006	20000215
0000000005	稼萌	女	19760612	160	50	001	004	20004372
0000000006	倪佳	女	19780102	170	70	001	003	20010220
0000000007	于卉	女	19780102	165	50	001	003	20004372
0000000008	張一風	女	19780311	159	50	001	003	20024372
0000000010	唐煜	女	19780501	160	50	001	003	20020606
0000000012	高逸菲	女	19810713	170	53	001	002	20030307
0000000015	徐陟	女	19830102	160	60	001	001	20050606
0000000016	周超	女	19830704	170	56	001	001	20050606

### 3.3.3 逻辑文件的编码范例

一般情况下，逻辑文件对应的源程序，与物理文件一样，都建立在专门放置数据库描述成员的源文件 QDDSSRC 中。

首先，我们要启动源语句输入实用程序（SEU）。在命令行键入 STRSEU，然后按 F4，配置参数，如下图：



Type choices, press Enter.

```
Source file . . . . . > QDDSSRC      Name, *PRV
Library . . . . . > TST010      Name, *LIBL, *CURLIB, *PRV
Source member . . . . . > SYMASTL1 Name, *PRV, *SELECT
Source type . . . . . > LF       Name, *SAME, BAS, BASP...
Option . . . . . > 2          *BLANK, ' ', 2, 5, 6
Text 'description' . . . . . > 雇员基本信息表逻辑文件 1
```

F3=Exit    F4=Prompt    F5=Refresh    F12=Cancel    F13=How to use this display  
F24=More keys

类型：我们建的是逻辑文件，所以，此处 LF

其他参数配置与物理文件相同,配置好后,回车进入 SEU 编辑界面:

```
Columns . . . : 1 80                      Edit TST010/QDDSSRC  
SEU==>                                     SYMASTL1  
FMT LF .....A.....T.Name+++++.Len++TdP.....Functions+++++  
*****Beginning of data *****  
  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
  
*****End of data *****  
  
F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle  
F16=Repeat find   F17=Repeat change   F24=More keys  
Member SYMASTL1 added to file TST010/QDDSSRC.
```

因为，生成的类型中，我们键入的是 LF 类型，所以，进入编辑界面后，画面显示的是 A 式样。在代码行处按 F4 键，可以查看代码格式提示。

Columns . . . : 1 80	Edit	TST010/QDDSSRC
SEU==>		SYMASTL1
FMT LF	.....A.....T.Name+++++.Len++TDpB.....Functions+++++	
	*****Beginni ngof data *****	
0001.00	A R SYMASTR	PFILE(SYMAST) 081013
0002.00	A K SYBMCD	081013
0003.00	A K SYSYKY	081013
0004.00	A S SYSYBT	COMP(EQ ' 女 ') 081013
	***** End of data *****	
<div> <div>Prompt type . . . LF</div> <div>Sequence number . . . 0004.00</div> <div> <div>Name</div> <div>Type</div> <div>S</div> <div>Functions</div> <div>COMP(EQ ' 女 ')</div> </div> <div> <div>Name</div> <div>SYSYBT</div> </div> <div>Length</div> <div>Data</div> <div>Type</div> <div>Decimal</div> <div>Posi tions</div> <div>Use</div> </div>		
F3=Exit	F4=Prompt	F5=Refresh
F12=Cancel	F23=Select prompt	F11=Previous record
		F24=More keys

根据上边的代码图，我们按从左到右的顺序讲解。

#### A 列:

与物理文件相同。

#### T 列:

类型项。不入力为在描述字段。有 4 种入力类型。

‘ R ’ 说明定义了一个记录格式，如第 1 行的 T 列代码，紧跟其后边的是记录名。

‘ K ’ 说明定义键值，如 2-3 行代码标记，此表内将按照此字段排序（默认为升序）。

‘ S ’ 说明定义抽出条件，如 4 行，紧跟后边的是抽出条件。同一个‘ S ’ 下定义的是并且关系。不同个‘ S ’ 下定义的是或者关系。

‘ O ’ 说明对象外数据条件，符合其后边定义条件的数据为不抽出数据。比较越少，工作效率越高，因此，当有几个选择/省略比较要处理时，应首先指定选择或忽略记录最多的比较。

‘ J ’ 说明连接逻辑文件，使用连接逻辑文件，应用程序只做一个读操作(对连接逻辑文件的记录格式)就可得到两个物理文件中所有数据。

#### NAME 列:

此列分三种情况:

一种为记录格式名。如第 1 行代码，说明的是此表的记录格式名为 SYMASTR。对于单格式和多格式逻辑文件，可以指定多个记录格式名，但在这个文件中每个名字必须是唯一的。记录格式名可以与在建立文件命令中指定的文件名字相同。然而，如果名字不唯一，将送出一个警告信息。象 RPG III 那样的一些高级语言处理程序不允许记录格式名与文件名相同。连接逻辑文件仅可以指定一个记录格式名。指定一个记录格式名作为逻辑文件中指定了字段的一个新的记录格式名。对于连接逻辑文件，在记录格式中的每一个字段必在 19-28 位中给出名字。只有在这个记录格式之外的其地方指定了这些字段名，物理文件的字段作为有 RENAME、CONCAT 和 SST 键字的参数时，这些字段才能是逻辑文件记录格式的一部分。JFILE 键字必须在记录层指定。它指定记录格式连接哪些物理文件。

第二种为字段名。如 2-3 行的代码，此处的含义为字段的定义名，或条件字段名。定义时，字段名不可以重复。如果不使用已经存在的物理文件中的记录格式，那么必须在逻辑文件中命名每个字段。在单格式或多格式逻辑文件中，在记录格式中的每个字段名必须是唯一的并且必须与在物理文件记录格式中的一个字段相对应。字段名的顺序是使用逻辑文件的程序中字段出现的顺序。在逻辑文件记录格式中出现的字段名通常是与在物理文件记录格式中相应的字段名相同。如果名字不同，那么这两个名字必须用 RENAME 键字使其相等。逻辑文件记录格式中的一个字段可以来自物理文件的两个或更多字段的连结。（CONCAT（连结）键字）。SST 键字可以用物理文件字段的子串作为逻辑文件的一个字段。

注：在逻辑文件中字段名顺序是重要的，如果在逻辑文件中一个记录格式中多次使用相同的物理文件字段，（使用 RENAME 或 CONCAT），那么字段在逻辑文件中顺序是数据传送到物理文件的顺序。那样，在逻辑文件中最后一次指定的字段值作为在物理记录中的值。

第三种键字段名。当第 17 列给出 K 时，19-28 列指定的名字是一个键字段名。它必须是物理文件的记录格式中字段名中的一个。这个字段的内容做为从数据库中检索记录的顺序。指定键字段是可选的。如果没有指定键字段，缺省的顺序是到达顺序（即这个记录输入到数据库时的顺序）。见数据库指南中关于访问路径的信息和它们在保存/重存操作时的影响。

使用这个键字段（可选地，选择/省略字段）在逻辑文件成员的记录格式中定义一个键字顺序访问路径。逻辑文件成员包括在建立逻辑文件（CRTLF）或增加逻辑文件成员（ADDLFM）命令中参数 DTAMBR 指定的物理文件成员。

通过指定顺序处理的键字，可以改变记录从文件中读出的顺序。这些顺序处理的键字是 ALTSEQ、NOALTSEQ、SIGNED、UNSIGNED、ABSVAL、ZONE、DIGIT、DESCEND、FIFO 和 LIFO。关于更多的信息参照相关书籍。

当没有对键字段指定顺序时，它的缺省顺序是升序的。

#### 数据类型 T 项：

数据类型。常用的有效的数据类型有：

- A          字符型
- P          压缩十进制
- S          区位十进制
- 0          全角半角混用，（注：当 0 型时，字段位数必须是偶数位）

其他类型，请参照物理文件中数据类型的描述，也可以把光标定位在数据类型处，按 F1 帮助查看。

#### FUNCTIONS 项：

功能定义，用于定义各种关键字

1. 文件级，位于记录名之前，主要包括：

UNIQUE：键值唯一（如本实例代码的第 1 行），定义后，数据（包括物理文件）不能有重复的键值数据，否则报错。

JFILE：联合物理文件用。

PFILE：指定使用的物理文件。

2. 记录级，位于记录名和第一个字段名之间，主要有

FORMAT：共享其他文件的记录格式

TEXT：记录格式说明

VALUE：将字段的内容和表中值相匹配（最多 100 个）的作为选择或省略记录。

3. 字段级，关键字位于一个字段名和其下一个字段名或第一个关键字段名之间，包括：

COLHDG：为数据库文件中字段显示和打印用

SST ：根据已定义字段，截取其数据位数定义。

4. 关键字段级，关键字位于第一关键字段名和其下一个关键字段名或成员尾之间，主要包括：

DESCEND：按降序排列，可用于字符或数值型键字段

ABSVAL：按绝对值顺序排序

更多键字，请参照物理文件 FUNCTIONS 项给出的数据文件有效键字表，也可以在代码处 F4, 然后光标定位到功能项处，F1 查看帮助说明。

在定义表过程中，可以在上边的命令行，键入 SAVE，保存当前。也可键入 FILE 命令，保存当前代码，退出。

### 3.3.4 建立逻辑文件

定义好源逻辑文件后，可以建立逻辑文件，来对数据进行承载。用 CRTLF 命令创建逻辑文件。在 PDM 显示界面命令行，键入 CRTLF（也可在源逻辑文件前 OPT 键入 14），F4, 配置参数如下：

Create Logical File (CRTLF)		
Type choices, press Enter.		
File . . . . .	> SYMASTL1	Name
Library . . . . .	> TST010	Name, *CURLIB
Source file . . . . .	> QDDSSRC	Name
Library . . . . .	> TST010	Name, *LIBL, *CURLIB
Source member . . . . .	> SYMASTL1	Name, *FILE
Generation severity level . . . .	20	0-30
Flagging severity level . . . .	0	0-30
File type . . . . .	*DATA	*DATA, *SRC
Member, if desired . . . . .	*FILE	Name, *FILE, *NONE
Physical file data members:		
Physical file . . . . .	*ALL	Name, *ALL
Library . . . . .		Name, *CURRENT
Members . . . . .		Name, *NONE
	+ for more values	
	+ for more values	
More...		
F3=Exit	F4=Prompt	F5=Refresh
F10=Additional parameters	F12=Cancel	
F13=How to use this display	F24=More keys	

F10 键增加参数设定，可以 PAGNUP, PAGEDOWN 上下翻页。



Create Logical File (CRTLF)

Type choices, press Enter.

Text 'description' . . . . . 雇员基本信息逻辑文件 1

---

Additional Parameters

Source listing options . . . . .		*SRC, *NOSRC, *SOURCE. . .
+ for more values		
System . . . . .	*LCL	*LCL, *RMT, *FILETYPE
Maximum members . . . . .	1	Number, *NOMAX
Access path size . . . . .	*MAX1TB	*MAX1TB, *MAX4GB
Access path maintenance . . . . .	*IMMED	*IMMED, *DLY, *REBLD
Access path recovery . . . . .		*NO, *AFTIPL, *IPL
Force keyed access path . . . . .	*NO	*NO, *YES
Preferred storage unit . . . . .	*ANY	1-255, *ANY
Rcd format selector program . . . . .	*NONE	Name, *NONE
Library . . . . .		Name, *LIBL, *CURLIB

More. . .

F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display  
F24=More keys

#### 参数说明

文件名：编译后生成的逻辑表名称。

库名：指定数据表生成库位置

源程序文件名：源代码所在文件名称。

源文件库：源代码所在库

源程序：源代码名

检查记录格式描述变更(LVLCHK)：在打开文件时，系统检验所用的文件记录格式自编译后是否改过，以至于程序不能处理文件。正常的系统会将上述情况通知程序，这种情况叫做级别检验。当使用生成或修改文件命令时，可以指定要做这个级别检验，也可以在 OVRDBF 命令中用 LVLCHK 参数覆盖文件级别检查。一般为了程序的健壮性，此处亦可以设定为\*NO。

其他参数，根据实际要求处理，一般为默认即可。

配置完后，回车执行。没错误，有成功信息提示；有错误，参照物理文件作成错误处理方法解决。

#### 3.3.5 其他种类逻辑文件介绍

前面，我们以单格式逻辑文件为例，讲解了逻辑文件的创建和生成。由于其他逻辑文件在实际项目开发中，应用的不是非常频繁。在本小节，我们对多格式逻辑文件以及连接逻辑文件举例简要说明。

##### 多格式逻辑文件

多格式逻辑文件既可以含有多个记录格式亦可在 PFILE 键字上指定多个文件。连接逻辑文件有一个记录格式并有 JFILE 键字，可以最多指出 32 个文件。下面给出多格式逻辑文件编码的例子。

```

|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8
00010A* LOGICAL FILE EXAMPLE
00020A* INVENTORY FORMAT
00030A      R INVFMT                PFILE(INVENTORY)
00040A      K ITEM
00050A*
00060A* ORDER FORMAT
00070A      R ORDFMT                PFILE(ORDER)
00080A      TEXT(' ORDER ANALYSIS' )
00090A      ITEM
00100A      ORDER          10
00110A      SUPPLY        +2
00120A      SHPDAT                CONCAT(SHPMO SHPDA SHPYR)
00130A      QTY          5P        RENAME(QTYDUE)
00140A      K ITEM
00150A      K SHPYR
00160A      K SHPMO
00170A      K SHPDA
00180A      O QTYDUE                CMP(LT 1)
00190A*
00200A* ACCOUNTING FORMAT
00210A      R ACTFMT                PFILE(ACCOUNTS)
00220A      FORMAT(ACCOUNTL)
00230A      K ITEM

```

对于 DDS 的使用和含义，请参照单格式逻辑文件中对于 DDS 的讲解，在这里就不再赘述。

### 连接逻辑文件

连接逻辑文件是把多个物理文件中的数据字段连接到一个记录格式中。对连接逻辑文件必须在记录层指定 JFILE 键字。在一个连接逻辑文件中仅允许有一个记录格式，所以这些项只可以指定一次。下图给出一个逻辑文件编码的例子：

```

00010A* 从两个物理文件把字段合并到一个记录格式
00020A      R RECORD1                JFILE(PF1 PF2)
00030A      J                        JOIN(PF1 PF2)
00040A      JFLD(NAME NAME)
00050A      NAME                      JREF(1)
00060A      ADDR

```

## 3.4 数据文件实用程序（DFU）使用与说明

### 3.4.1 DFU 简介

数据文件实用程序（DFU）是一个程序生成器，帮助用户生成一个程序来输入数据、更新文件和查询文件，不用使用程序设计语言就可以用 DFU。由响应几个显示就可以建立 DFU 程序。DFU 用临时程序提供给用户快速更新文件的方法，而不需首先定义 DFU 程序，它也能快速生成数据库管理程序，比用程序设计语言要快。本节将根据实例介绍 DFU 的常用功能。

可在命令入口显示中用 STRDFU 命令或在命令行写一个有参数的命令来启动 DFU，可用 STRDFU



命令定义，修改及运行 DFU 程序。如果有权力的话，也可直接用 CHGDTA、DSPDTA 或 UPDDTA 来运行已有的程序或临时程序。

在命令行键入 STRDFU, 执行。出现 DFU 的主菜单：

AS/400 Data File Utility (DFU)

Select one of the following:

1. Run a DFU program

2. Create a DFU program

3. Change a DFU program

4. Delete a DFU program

5. Update data using temporary program

Selection or command

====> \_\_\_\_\_

F3=Exit    F4=Prompt    F9=Retrieve    F12=Cancel

(C) COPYRIGHT IBM CORP. 1981, 2002.

DFU Option 有 5 中选择，依次为：1、运行 2、新建 3、改变 4、删除 5、运行临时 DFU 程序。

#### 3.4.2 运行 DFU

当系统中存在 DFU 对象时，我们就可以运行它。例如，在 TST010 中，存在 SYMAST 的对象 TESTDFU，我们要运行它，就在 DFU 的主菜单的命令行处，键入 1，执行，选择 2（显示），执行，配置参数下图：

Change a Data File		
Type choices, press Enter.		
Program . . . . .	<u>TESTDFU</u>	Name, F4 for list
Library . . . . .	<u>TST010</u>	Name, *LIBL, *CURLIB
Data file . . . . .	<u>SBMAST</u>	Name, *SAME, F4 for list
Library . . . . .	<u>TST010</u>	Name, *LIBL, *CURLIB
Member . . . . .	<u>SBMAST</u>	Name, *FIRST, F4 for list
F3=Exit      F4=Prompt      F12=Cancel		

执行后，即显示了 TESTDFU 的内容。其中的数据来至于 SYMAST，显示的形式是在建立 TEETDFU 时定义的。

类似的操作，我们也可以变更（3）和删除（4）已有的 DFU 对象。

### 3.4.3 创建 DFU

我们也可以创建一个 DUF 对象，用于编辑或显示数据表。例如，我们要为 SBMAST 表建立一个 SBDFU 的 DFU 程序，在 DFU 主菜单，键入命令 2，执行

### 建立 DFU 程序

输入选项，按“ 执行” 键。

程序 . . . . .	<u>SYDFU</u>	名称, F4 显示列表
库 . . . . .	<u>TST010</u>	名称, *CURLIB
数据文件 . . . . .	<u>SYMAST</u>	名称, F4 显示列表
库 . . . . .	<u>TST010</u>	名称, *LIBL , *CURLIB

F3= 退出      F4= 提示      F12= 取消

在执行, 进入定义一般信息界面

### 定义一般信息 / 索引的文件

输入选项，按“ 执行” 键。

作业标题 . . . . .	<u>SYDFU</u>	
显示格式 . . . . .	<u>2</u>	1= 单个, 2= 多个 3= 最大, 4= 行定向
审计报告 . . . . .	<u>N</u>	Y= 是, N= 否
S/36 类型 . . . . .	<u>N</u>	Y= 是, N= 否
抑制错误 . . . . .	<u>N</u>	Y= 是, N= 否
编辑数字 . . . . .	<u>Y</u>	Y= 是, N= 否
在卷动时容许更新 . . . . .	<u>Y</u>	Y= 是, N= 否
键:		
生成 . . . . .	<u>N</u>	Y= 是, N= 否
允许更改 . . . . .	<u>Y</u>	Y= 是, N= 否

F3= 退出      F12= 取消      F14= 显示定义

按上图定义完毕后，回车执行。



## 使用记录格式

文件 . . . : SYMAST

库 . . . . . : TST010

输入选项, 按“ 执行” 键。按 F21 选择全部。

2= 指定 4= 删除

Opt	格式	定义的	描述
—	SYMASTR	N	

Bottom

F3= 退出

F5= 刷新

F12= 取消

F14= 显示定义

F21= 选择全部

按执行键进入下一屏, 以选择 2 选定数据格式。

## 选择和序列字段

文件 . . . . . : SYMAST  
记录格式 . . . . . : SYMASTR

库 . . . . . : TST010

选择字段及其顺序, 或按 F21 选择全部; 按“ 执行” 键。

顺序	字段	属性	长度	类型	描述
_____	SYSYCD	KEY	10	CHAR	雇员 I D
_____	SYSYNM		20	DBCS	雇员名
_____	SYSYBT		4	DBCS	性别
_____	SYSNDT		8,0	ZONE	出生年月日
_____	SYSNTY		3,0	ZONE	身高
_____	SYTAYX		3,0	ZONE	体重
_____	SYBMCD		3	CHAR	部门 I D
_____	SYSYKY		3	CHAR	职位
_____	SYNYDT		8,0	ZONE	入社日期

Bottom

F3= 退出

F5= 刷新

F12= 取消

F14= 显示定义

F20= 重新编号

F21= 选择全部

按 F21 选定所有数据项, 然后执行进入下一屏:

使用字段			
文件 . . . . . :	SYMAST	库 . . . . . :	TST010
记录格式 . . . . . :	SYMASTR		
输入选项, 按“ 执行” 键。按 F21 选择全部。			
2= 指定扩充的定义			
4= 删除扩充的定义			
Opt	字段	扩充的 定义	标题
—	SYSYCD	N	雇员 I D
—	SYSYNM	N	雇员名
—	SYSYBT	N	性别
—	SYSNDT	N	出生年月日
—	SYSNTY	N	身高
—	SYTAYX	N	体重
—	SYBMCD	N	部门 ID
—	SYSYKY	N	职位
			More...
F3= 退出		F5= 刷新	F12= 取消
F14= 显示定义		F21= 选择全部	

如要编辑日期的显示格式, 在日期前入力 2, 执行

指定扩充的字段定义			
字段 . . . . . :	SYSNDT	记录格式 . . . . . :	SYMASTR
输入选项, 按“ 执行” 键。			
自动复制 . . . . .	<u>N</u>	Y= 是, N= 否	
累加 . . . . .	<u>N</u>	Y= 是, N= 否	
扩充的字段			
标题 . . . . .	出生年月日		
标题位置 . . . . .	*BEFORE	*ABOVE, *BEFORE	
初始值 . . . . .			
自动增量 . . . . .			
有效性检查 . . . . .	2= 更改, 4= 删除		
			More...
F3= 退出	F12= 取消	F14= 显示定义	

定义自动复写是否自动复制, 可以节省录入时间; 定义累算可以再录入结束之后给出累计数值; 可以自定义数据项标题。在本屏幕使用 PageDown 下翻一页; 定义字段项扩展选择, 这里是改变了 SYSNDT 字段的编辑码, 在编辑字处, 编辑显示格式。再键入 2 次执行, 参见以下图表, 为即将生成的 DFU 程序。再键入 2 次执行, 参见以下图表, 为即将生成的 DFU 程序。

退出 DFU 程序定义		
输入选项，按“ 执行” 键。		
保存程序 . . . . .	<u>Y</u>	Y= 是, N= 否
运行程序 . . . . .	<u>Y</u>	Y= 是, N= 否
对于选择“ Y= 是”:		
运行的类型 . . . . .	<u>1</u>	1= 更改, 2= 显示
修改程序 . . . . .	<u>N</u>	Y= 是, N= 否
保存 DDS 源 . . . . .	<u>N</u>	Y= 是, N= 否
对于保存程序“ Y= 是”:		
程序 . . . . .	<u>SYDFU</u>	名称
库 . . . . .	<u>TST010</u>	名称, *CURLIB, . . .
权限 . . . . .	<u>*LIBCRTAUT</u>	名称, *LIBCRTAUT, . . .
文本 . . . . .	<u>SYDFU</u>	
对于保存 DDS 源“ Y= 是”:		
源文件 . . . . .		名称
库 . . . . .	<u>*CURLIB</u>	名称, *CURLIB, . . .
源成员 . . . . .	<u>SYDFU</u>	名称
F3= 退出	F14= 显示定义	F17= 快速路径

执行后，进入录入数据画面

Change a Data File		
Type choices, press Enter.		
Program . . . . .	<u>SYDFU</u>	Name, F4 for list
Library . . . . .	<u>TST010</u>	Name, *LIBL, *CURLIB
Data file . . . . .	<u>SYMAST</u>	Name, *SAME, F4 for list
Library . . . . .	<u>TST010</u>	Name, *LIBL, *CURLIB
Member . . . . .	<u>*FIRST</u>	Name, *FIRST, F4 for list
F3=Exit F4=Prompt F12=Cancel		
The DFU program was saved successfully.		

再次键入执行后，即可按下边的帮助信息，对数据进行处理。也可 F3，退出 DFU,此时，在生成的库下就可以看到生成的两个 TBDFU 对象（一个属性 FILE,一个属性 DUF）了。与运行临时 DFU 不同的是，使用 STRDFU 生成的 DFU 程序，可以反复调用，并嵌入到程序里使用。



#### 3.4.4 运行临时 DFU 程序

是我们常用的编辑数据的方法。在 DFU 主菜单，我们可以键入 5，执行，键入要更新的表和其所在的库后再次执行。也可通过在 OS/400 命令行键入 UPDDTA+F4，然后分别在 Data base file、Library 输入数据库文件名、对应库名来快速使用 DFU。此种方法，一旦修改结束，程序随之消除。

到这里，我们有关 AS400 的数据库的基础部分就介绍完了，关于数据库在程序中的应用在后面的部分会介绍到。

## 第四章 控制语言（CL）

### 4.1 基本概念

CL 程序就是和RPG 相对应的，是控制语言（Control Language）。类型为CLP、CLLE 的源代码编译出来的程序，都属于CL 程序。

CL 程序不像RPGLE 程序，在编写时，可以使用自由格式书写；一行的内容如果太长要，在最末尾处用“+”表示换行

举个最简单的例子，比如说新建个名为FHS01CL 的CLP 源程序，代码如下：

```
FMT ** ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
***** データの始*****
0001.00          PGM
0002.00 /*      */
0003.00          WRKACTJOB
0004.00 /*      */
0005.00          ENDPGM
***** データの終*****
```

编译此程序（一是，使用 CL 命令 CRTCLPGM，二是，使用PDM 菜单选项 14），然后CALL 之，系统就会执行命令WRKACTJOB，查看当前的活动作业，效果与在交互式命令行下输入WRKACTJOB 是一样的。当我们输入F12，退出WRKACTJOB 时，系统就会继续向下执行，发现是ENDPGM，表示程序结束了，于是判定执行完毕，退出至交互式画面。

### 4.2 CL编程

#### 4.2.1 程序的开始与结束

```
FMT ** ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
***** データの始*****
0001.00          PGM      PARM(&A &B)          /* 开始CL 程序 */
0002.00 /*      */
0003.00          WRKACTJOB
0004.00 /*      */
0005.00          ENDPGM          /* 结束CL 程序 */
***** データの終*****
```

CL 程序，和RPGLE 程序一样，也可以有程序的入口参数，而且程序的入口参数都是可传递的（也就是输入的参数如果在程序中被修改过，那么原调用的程序中的相应参数也会进行变化。不过CL 的入口参数只能为字符型，或数字型的单个字段，不能象RPGLE 程序中那么多多样化（字段、结构、数组、指针）。

如果CL 程序没有入口参数时，那么就可以不需要后面的PARM 语句，直接写成PGM即可。

写CL 程序时，不妨多使用F4，看看系统的帮助，这样就不用记那么多命令的参数名。

#### 4.2.2 变量及其定义

CL程序中的所有变量，都使用&做为前缀，这一点与RPG 程序不同。比如说

PGM PARM (&A &B) 就表示入口参数为A、B 这两个变量

```
FMT ** ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
***** データの始め *****
0001.00          PGM      PARM(&FLD01 &FLD02)
0001.01 /*      */
0001.02          DCL VAR(&FLD01) TYPE(*CHAR) LEN(10)
0001.03          DCL VAR(&FLD02) TYPE(*DEC)  LEN(10 2)
0002.01 /*      */
```

在CL 程序中使用到的变量，都必须使用DCL 语句来定义：

定义变量FLD01，10 位长的字符型变量，定义变量FLD02，10 长，其中2 位小数的数字型变量。

除了字符、数字之外，CL 程序还可以定义逻辑变量(\*LGL)，逻辑变量允许的值只能为' 1' 或' 0' 。不过通常有字符与数字也就够了。CL 程序的主要功能在于进行命令处理，而不是处理字符串以及数

```
FMT ** ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
***** データの始め *****
0001.00          PGM      PARM(&FLD03)
0001.01 /*      */
0001.04          DCL VAR(&FLD03) TYPE(*LGL) VALUE(0)
0002.01 /*      */
```

数据库。

#### 4.2.3 CL常用命令

##### 1. CHGVAR -- 变量赋值

CHGVAR VAR(&FLD01) VALUE(' ABCD' )即是将变量FLD01 赋值成为' ABCD' (左对齐)，同理，VALUE 的括号中也可以填写一个变量，即表示将此变量的值赋值到变量FLD01 中。数字型变量的赋值同样也是使用CHGVAR 语句。

当变量中只包含数字时（0—9），数字型变量与字符型变量可以使用CHGVAR语句进行转换，这一点与RPGLE 中的MOVE 语句比较类似。

## 2. IF -- 条件判断语句

```

FMT ** ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
***** データの始め *****
0001.00          PGM      PARM(&FLD01 &FLD02)
0001.01 /*      */
0001.02          DCL VAR(&FLD01) TYPE(*CHAR) LEN(10)
0001.03          DCL VAR(&FLD02) TYPE(*DEC)  LEN(10 2)
0002.01 /*      */
0002.02          IF          COND(&FLD01 *EQ '1') THEN(CHGVAR VAR(&FLD02) +
0002.03          VALUE('0'))
0002.04 /*      */
0003.00          ENDPGM
***** データの終り *****

```

当变量FLD01 等于' 1' 时，将变量FLD02 中的值更改为' 0' THEN 后面，即是当符合条件时，要执行的命令。写起来其实没有看上去那么复杂，多用F4 就会发现CL 程序写简单。

If (IF)	
Type choices, press Enter.	
Label	.....
Condition	..... > (&FLD01 *EQ '1')
Command	..... > CHGVAR VAR(&FLD02) VALUE('0')
Comment	.....
Bottom	
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display	
F24=More keys	

就如如上例，先写IF，然后按F4，在Condition 处填写条件语句，然后在Command 处填上CHGVAR，再按F4，再去填相应的处理语句，这样写，就比直接把整句抄下来就简单多了。

上面这种写法，只能在符合条件时，执行一条CL 语句；如果要执行多条，就必须写做：

```

IF COND(&FLD01 *EQ '1') THEN(DO)
CHGVAR VAR(&FLD02) VALUE('0')

```

其它执行语句

ENDDO

也就是THEN 后面，用DO，表示接下来的语句都是在这个IF 条件成立时才执行（DO）的。然后结束处用ENDDO，必须要有。ENDDO 在这里和循环没有任何关系，表示的是ENDIF 的意思，但是CL 语句里没有ENDIF，只有ENDDO。IF 语句中，表示判断的关键字与RPGLE 中的Ifxx 操作码类似，有

\*EQ \*GT \*LT \*GE \*LE \*NE

用来表示逻辑关系的关键字有

\*AND, \*OR, \*NOT

### 3. GOTO -- 跳转语句

GOTO 语句与RPGLE 中的GOTO 是一样的，都是跳转的意思。

FHSTAG: GOTO CMDLBL(FHSTAG)

注意，这里定义标签是用“ : ” 冒号

### 4. MONMSG -- 监控错误信息

我们使用CL 语句时，执行的命令可能会报出一些异常错误，从而导致整个程序中断，需要手工干预。MONMSG 命令可以监控我们预定的错误信息，使CL 程序正常的向下运行。举例而言，如果CL 程序中有如下语句：

CALL PGM(FHS01R)

MONMSG MSGID(CPF4131)

则表示当系统调用程序FHS01R 时，如果发现有CPF4131（声明的文件重新编译过，但程序未重新编译）的错，那么CL 程序将不会异常中断，仅仅只是不运行程序FHS01R，然后继续向下执行CL 程序MONMSG 还可以用于在监控到错误信息之后，进行处理，如下：

CALL PGM(FHS01R)

MONMSG MSGID(CPF4131) EXEC(CHGVAR VAR(&FLD01) +  
VALUE(' 0' ))

这句话就表示当发现有CPF4131 的错误之时，将变量FLD01 赋值成为' 0' 如果要执行多句的话，和IF 语句的方法类似，也是使用DO 与ENDDO

MONMSG MSGID(CPF4131) EXEC(DO)

CHGVAR VAR(&FLD01) VALUE(' 0' )

其它处理语句

ENDDO

### 5. %SST -- 取字符串

CHGVAR VAR(&FLD01) VALUE(%SST(&FLD02 3 1))

表示将字符变量FLD02，从第3 位开始，取1 位，左对齐赋值到变量FLD01 中。%SST 的括号的

参数中，第一个参数必须为字符型变量，第二个参数表示起始位，第三个参数表示要截取的长度。

#### 6. \*CAT -- 拼字符串

```
DCL VAR(&FLD01) TYPE(*CHAR) LEN(10)
```

```
CHGVAR VAR(&FLD01) VALUE('A' *CAT 'B')
```

即表示将10 位长的字符型变量赋值成为 ' AB '

' A' ， ' B' ， 也可以使用变量，如

```
CHGVAR VAR(&FLD01) VALUE(&FLD02 *CAT &FLD03)
```

要注意，\*CAT 不能去掉字符串末尾的空，从效果上来看，有点类似于RPGLE 中的EVAL 操作码，而不是CAT 操作码

#### 7. +、-、\*、/ -- 数学运算

数字型变量，可以进行数学运算

```
CHGVAR VAR(&FLD01) VALUE(&FLD01 + &FLD02)
```

即等同于RPGLE 程序中的 EVAL FLD01 = FLD01 + FLD02

同理，-、\*、/ 分别对应减、乘、除

不过数学运行常用于RPGLE 程序中，较少用在CL 控制里面，这里只是介绍一下。

#### 8. 读取文件：（From Cuer: P1421）

```
DCL VAR(&FLD01) TYPE(*CHAR) LEN(2)
```

```
DCLF FILE(FHSLIB/PFFHS)
```

```
RCVF
```

```
CHGVAR VAR(&FLD01) VALUE(&FHS01)
```

以上这段CL 的意思，就是在CL 程序中读取PFFHS 文件，然后将读到的第一条记录赋值到CL 的临时变量FLD01 中。

如果要将一个文件从头读到尾，则可以用如下语句来实现：

```
DCLF FILE(FHSLIB/PFFHS)
```

LOOP:

```
RCVF
```

```
MONMSG MSGID(CPF0864) EXEC(GOTO CMDLBL(EXIT))
```

读取到每条记录后的处理语句

```
GOTO CMDLBL(LOOP)
```

EXIT:

也就是说，信息CPF0864，即表示未读取到记录。

在CL 程序中声明文件使用DCLF 语句，一个CL 文件中只能声明一个文件，声明语句必须在CL 控制语句之前。使用声明的文件中的字段，同样需要在字段名前加上“ & ” ；

CL 程序中，无法控制游标，无法对记录进行定位操作。所以CL 程序对文件的操作是比较弱的，

通常我最多只用来读取某些只含少量记录的参数文件。

#### 4.2.4 CL 过程的组成部分

做为 CL 过程，每部分的源语句实际上都是一个 CL 命令，在典型的 CL 过程中它可以分成下列几个基本部分：

PGM 命令	PGM PARM(&A) 这是一个可选值，表示过程的开始及可接收的参数
说明命令	(DCL, DCLF) 在使用变量时，说明命令要放在除 PGM 命令之外的所有命令前面
CL 处理命令	CHGVAR, SNDPGMMSG, OVRDBF, DLTF..... 用来管理常量及变量
逻辑控制命令	IF, THEN, ELSE, DO, ENDDO, GOTO 用来控制过程内部的处理
内部函数	%SUB, %SWICH, %BIN 用于计算、关系或逻辑表达式
程序控制命令	CALL, RETURN 用来把控制传递给其它程序
过程控制命令	CALLPRC, RETURN 用来把控制传递给其它过程
ENDPGM 命令	ENDPGM 这是可选的，表示程序的结束

以上命令使用的顺序、组合及扩展都依据应用程序的逻辑和设计。在生成过程、处理命令时，引用的对象必须已经存在。

#### 4.2.5 CL 过程中使用的命令分组说明

系统功能	命 令	命令功能
修改过程控制	CALL (Call)	调用一个程序
	CALLPRC (Call Procedure) (1)	调用一个过程
	RETURN (Return)	返回到引起程序或过程运行的下一条命令
CL过程界限	PGM (Program) (1)	指出CL过程源码的开始
	ENDPGM (End Program) (1)	指出CL过程源码的结束
CL过程逻辑	IF (If) (1)	根据逻辑表达式的值执行命令
	ELSE (Else) (1)	对IF命令为假条件定义采取的动作
	DO (Do) (1)	指出DO组的开始
	ENDDO (End Do) (1)	指出DO组的结束
	GOTO (Go To) (1)	转移到另外的命令
CL过程变量	CHGVAR (Change Variable) (1)	修改CL变量的值
	DCL (Declare) (1)	说明一个变量
替换	CHGVAR (Change Variable) (1)	修改CL变量的值
	CVTDAT (Convert Date) (1)	修改日期格式
数据区	CHGDTAARA (Change Data Area)	修改数据区
	CRTDTAARA (Create Data Area)	生成一个数据区
	DLTDTAARA (Delete Data Area)	删除一个数据区
	DSPDTAARA (Display Data Area)	显示一个数据区
	RTVDTAARA (Retrieve Data Area)	把数据区的内容复制到一个CL变量中
文件	ENDRCV (End Receive) (1)	取消由前面的RCVF, SNDF或SNDRCVF命令对一个显示文件发出的输入请求
	DCLF (Declare File) (1)	说明一个显示文件或数据库文件
	RCVF (Receive File) (1)	从显示文件和数据库文件中读记录
	RTVMBRD (Retrieve Member Description) (1)	取得数据库文件成员的描述
	SNDF (Send File) (1)	往显示文件中写记录
	SNDRCVF (Send/Receive File) (1)	往显示文件中写记录, 在用户回答后读记录



	WAIT (Wait) (1)	等待从显示文件发出的SNDF, RCVF或SNDRCVF命令接收数据
信息	MONMSG (Monitor Message) (1)	监控送往程序信息队列的逃逸、状态和通知信息
	RCVMSG (Receive Message) (1)	把信息从信息队列复制到一个CL变量中
	RMVMSG (Remove Message) (1)	从信息队列取消信息
	RTVMSG (Retrieve Message) (1)	把预先定义的信息从信息文件复制到CL变量中
	SNDPGMSG (Send Program Message) (1)	往信息队列发送程序信息
	SNDRPY (Send Reply) (1)	给查询信息的发送者发送回答信息
	SNDUSRMSG (Send User Message)	给显示工作站或系统操作员发送消息或查询信息
混杂命令	CHKOBJ (Check Object)	检查对象是否存在及使用对象必须有的权限
	PRTCMDUSG (Print Command Usage)	产生一个用在某组CL过程中的一组命令中的交叉引用表
	RTVCFGSRC (Retrieve Configuration Source)	对生成的已存在的配置对象建立一个CL命令源码且把它放在源文件成员中
	RTVCFGSTS (Retrieve Configuration Status) (1)	从三个配置对象（线路、控制器和设备）中取得配置状态
	RTVJOBA (Retrieve Job Attributes) (1)	取得一个或多个作业属性的值且把它们放到CL变量中
	RTVSYSVAL (Retrieve System Value) (1)	取得系统值并且把它放到一个CL变量中
	RTVUSRPRF (Retrieve User Profile) (1)	取得用户配置文件属性并把它放到CL变量中
程序生成命令	CRTCLMOD (Create CL Module)	生成一个CL模块
	DLTMOD (Delete Module)	删除一个模块
	DLTPGM (Delete Program)	删除一个程序
	CRTBNDCL (Create Bound Control Language Program)	生成一个联编的CL程序
	CRTPGM (Create Program)	生成一个程序

	CRTSRVPGM (Create Service Program)	生成一个服务程序
--	------------------------------------	----------

#### 4.2.6 变量的使用

##### 4.2.6.1 变量说明

使用 DCL 命令来定义变量及其属性、长度和初值。标准格式如下：

```
DCL VAR (变量名) TYPE 

|        |
|--------|
| * CHAR |
| * DEC  |
| * LGL  |

 LEN (长度) VALUE(初值)
```

使用 DCL 命令时，必须遵循的规则如下：

1. CL 变量名必须以 & 开始，后跟字符不多于 10 个，& 后的第一个字符必须是字母，其余的可以是字母或数字。
2. CL 变量值必须是：字符型最长为 9999 个字符；数值型为压缩十进制数，最长为 15 位，其中小数位最长为 9 位；逻辑型为“0”或“1”，表示假或真。
3. CL 变量的缺省初值为：字符型为空，数值型为 0，逻辑型为“0”。
4. 对于字符型和数值型，如果指定了初值而未指定长度，则缺省长度为初值长度。

##### 4.2.6.2 变量值中小写字符的限制

用作变量的保留值必须用大写字母表示，特别是当它们用引号括起来的时候。例如：

```
DCL VAR(&LIB) TYPE(*CHAR) LEN(10) VALUE(' *LIBL' )
DLTPGM &LIB/MY PROG
```

注意：如果 VALUE 参数不使用引号，则小写是正确的。因为使用 SEU 编辑会自动转换成大写。

##### 4.2.6.3 变量赋值

使用 CHGVAR 命令给变量赋值，其值可改变成：

1. 常量： CHGVAR VAR (&A) VALUE (0)
2. 变量： CHGVAR VAR (&A) VALUE (&B)
3. 计算量： CHGVAR VAR (&A) VALUE (&A+1)
4. 函数量： CHGVAR VAR (&A) VALUE(%SST(&B 1 5))  
CHGVAR VAR(%SST(&A 1 5)) VALUE(&B)

赋值时应注意以下几点：

1. 对逻辑变量，被改变的值必须是一个逻辑值。
2. 对数值变量，只能赋予十进制数，或者数字字符变量（包括小数点和正负号）。
3. 对字符变量，既可接受字符，也可接受十进制数。赋十进制数时，该字符变量的值是右对齐，前导补零，负号放在最左边。

##### 4.2.6.4 在 CL 过程中写注释

如果是在 CL 过程中写注释或对命令加注释，使用字符对 /\* 和 \*/。注释放在这对符号中间。

注释的起始界限符/\*，要三个字符，除非/\*出现在命令串的头二个位置。此时，在命令前/\*后

不用跟空格。

可用下列方法之一写三个字符的注释起始界限（b 为空格）：

/\*b

b/\*

/\*\*

这样，注释起始界限符可由四种方法输入：

从命令串的第一个位置开始

前加一空格

后跟一空格

后跟一个星号（/\*\*）

在下面的过程中，写一个注释来提示用户可以响应的菜单选项：

```
PGM /* ORD040C Order dept general menu */
DCLF FILE(ORD040CD)
START:  SNDRCVF RCDfmt(MENU)
        IF (&RESP=1) THEN(CALL CUS210)
        /*Customer inquiry */
        ELSE +
            IF (&RESP=2) THEN(CALL ITM210)
            /**Item inquiry */
            ELSE +
                IF (&RESP=3) THEN(CALL CUS210)
                /* Customer name search */
                ELSE +
                    IF (&RESP=4) THEN(CALL ORD215)
                    /** Orders by cust */
                    ELSE +
                        IF (&RESP=5) THEN(CALL ORD220)
                        /* Existing order */
                        ELSE +
                            IF (&RESP=6) THEN(CALL ORD410C)
                            /** Order entry */
                            ELSE +
                                IF (&RESP=7) THEN(RETURN)

        GOTO START
ENDPGM
```

#### 4.2.7 CL 程序内部的逻辑控制

CL 过程中的命令是按顺序处理的，处理完一个再处理后面的一个。可用修改逻辑流程的命令来改变这个处理的顺序，这些命令可以是条件的（IF）或无条件的（GOTO）

无条件转移意思是不管转移指令发生时存在什么样的条件，过程执行都转移到过程中的某个命令或一组命令中，这时要用 GOTO 命令。

条件转移是在一定的条件下，处理要转移到过程内非连续的段或命令中，可转移到过程中的任何语句。因为转移发生在某个条件为真时，所以叫做条件处理。这时经常用 IF 命令，如果条件不为真，可用 ELSE 命令规定替代的处理。DO 命令生成一组一起处理的命令，即在某种条件下，执行一组命令。

#### 4.2.7.1 使用 GOTO 命令及标号

GOTO 命令处理无条件转移，无论什么时候遇到 GOTO 命令，都要直接执行过程中的另一部分（用一个标号指出），这种转移不依赖表达式的求值。转移到标号指定的语句后，从这个语句开始，继续顺序处理。它不返回到 GOTO 命令处，除非由另外指令特别指定返回。可以往前或向后转移，不能用 GOTO 语句走到过程外的标号处。GOTO 命令没有参数，它只包含要转移语句的标号：

GOTO CMDLBL(标号)

标号标识过程中的语句，指出 GOTO 命令执行的方向，要使用 GOTO 语句，要转移的去处必须有一个标号：

```
PGM
.
.
START:  SNDRCVF RCDfmt(MENU)
        IF (&RESP=1) THEN(CALL CUS210)
.
.
        GOTO START
.
.
ENDPGM
```

此例中，标号是 START，标号最多 10 个字符，结尾必须有冒号，命令名和标号之间有空格。

#### 4.2.7.2 使用 IF 命令

IF 命令用来设置一个条件。如果为真，要规定在过程中运行的语句或一组语句，与 IF 一起可以规定 ELSE 命令，来规定条件不为真时执行的语句。

命令包括一个表达式（它用来测试真、假）和一个 THEN 参数，它现定表达式为真时要采取的动作，IF 命令的格式为：

IF COND(逻辑表达式) THEN(CL 命令)

如果逻辑表达式的条件为真，处理 THEN 参数中的命令，它可以是一个命令，也可以是一组命令，如果条件不为真，运行下一个顺序的命令。COND 和 THEN 都是命令的键字，它们可以省略。下面是

正确的写法:

```
IF COND(&RESP=1) THEN(CALL CUS210)
IF (&A *EQ &B) THEN(GOTO LABEL)
IF (&A=&B) GOTO LABEL
```

在IF和键字或值(&A~~zz~~)之间要有空格,在键字之间不允许有空格,如果有,用左边的括号括住值。

下面是IF的例子,假定在编码前,&A的值为2,&4的值为4。

```
IF (&A=2) THEN(GOTO FINAL)
IF (&A=3) THEN(CHGVAR &C 5)
.
.
.
```

```
FINAL: IF (&C=5) CALL PROGA
ENDPGM
```

在这种情况下,处理第一个IF之后转到FINAL处,跳过中间的语句,不返回第二个IF命令处,在FINAL中,由于要测试&C是否为5,此时为假,则不调用PROGA,而执行ENDPGM。它指出过程结束,控制返回给调用它的过程。

如果变量的初值不同,则用同样的编码,处理逻辑会不同,比如开始&A为3,&C为4,则第一个IF为假,则不执行GOTO,而顺序执行第二个IF。测试为真,则把&C改为5。继续执行后续语句,当做到最后一个IF,此时&C=5,为真,则调用PROGA。

几个相邻IF语句是独立运行的,例如:

```
PGM /* IFFY */
DCL &A.
DCL &B.
DCL &C.
DCL &D.
DCL &AREA *CHAR LEN(5) VALUE(YESNO)
DCL &RESP..
IF (&A=&B) THEN(GOTO END) /* IF #1 */
IF (&C=&D) THEN(CALL PGMA) /* IF #2 */
IF (&RESP=1) THEN(CHGVAR &C 2) /* IF #3 */
IF (%SUBSTRING(&AREA 1 3) *EQ YES) THEN(CALL PGMB) /* IF #4 */
CHGVAR &B &C
.
.
.
END: ENDPGM
```

在上例中，如果 $\&A \neq \&B$ ，运行下面语句，假如 $\&C = \&D$ ，则调用 PGMA，当 PGMA 返回时，考虑第三个 IF，等等。

一个嵌套的命令是完全包括在另一个命令的参数中，在下例中，CHGVAR 和 DO 命令是嵌套的：

```
IF (&A *EQ &B) THEN(CHGVAR &A (&A+1))
    IF (&B *EQ &C) THEN(DO)
        .
        .
        .
    ENDDO
```

#### 4.2.7.3 使用 DO 命令和 DO 组

DO 命令处理一组命令，这一组命令是用 DO 和 ENDDO 之间的命令组成的，这组命令的处理通常是以有关命令为条件的。它经常与 IF、ELSE 或 MONMSG 一起使用。例如，

```
IF (&A=&B) THEN(DO)
    z
    z
    z
    ENDDO
z
z
z
ENDPGM
```

} DO 组

如果逻辑表达式 $(\&A = \&B)$ 为真，则执行 DO 组，如果不为真，则跳过 DO 组，处理 ENDDO 后的命令。

在下例中，如果 $\&A \neq \&B$ ，则调用 PGMB，不调用 PGMA，否则执行 DO 组中的命令。

```
IF (&A=&B) THEN (DO)
    CALL PGMA
    CHGVAR &A &B
    SNDPGMMSGzzz
    ENDDO
    CALL PGMB
    CHGVAR &ACCTS &B
```

DO 组

DO 组也能嵌入到另外的 DO 组中，最多可嵌套 10 层，下例中有三层，注意每个 DO 组都由 ENDDO 结束。

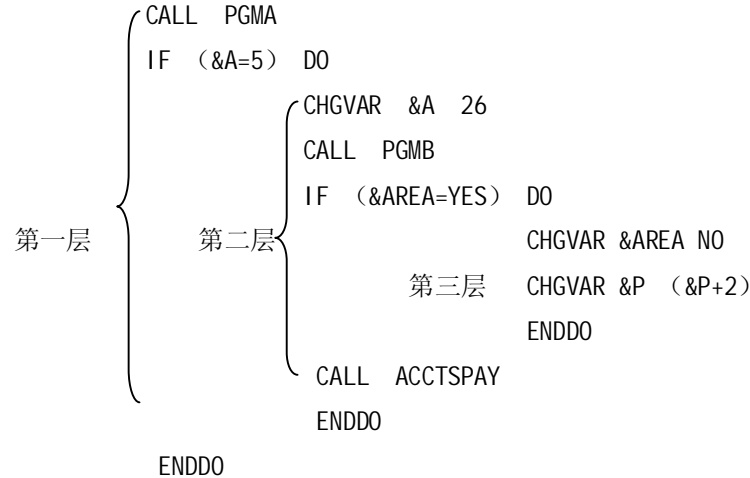
PGM

Z

Z

Z

IF (&amp;A=&amp;B) DO



CALL PGMC

ENDPGM

此例中，如果第一层的&A≠5，调用 PGMC，如果&A=5，则执行第三个 DO 组中的语句，如果第二个 DO 组中的&AREA≠YES，则调用 ACCTSPAY，这是国为处理转到 DO 组后的下一个命令。

CL 编译程序不指出 DO 组的起始和结束，如果编译程序发现不成对的条件，它不是很容易指出实际的错误，怎样指出 DO 组的层次错误请看 QUSRT00L 的 DSPCLPDO 中的例子。

#### 4.2.7.4 使用 ELSE 命令

ELSE 命令在 IF 命令的条件为假时规定所做的处理。

IF 命令也能不用 ELSE:

IF (&amp;A=&amp;B) THEN(CALLPRC PROCA)

CALLPRC PROCB

在这种情况下，仅在&A=&B时调用PROCA，而总是调用PROCB。

如果要在在这个过程中用 ELSE 命令，处理逻辑要修改，在下例中，如果&A=&B，则调用 PROCA，但不调用 PROCB，如果&A≠&B，则调用 PROCB。

IF (&amp;A=&amp;B) THEN(CALLPRC PROCA)

ELSE CMD(CALLPRC PROCB)

CHGVAR &amp;C 8

如果 IF 表达式有一个不为真条件导致明显的转移（即绝对的非此即彼转移），则必须用 ELSE 命令。

实际上 ELSE 非常有用体现在与 DO 组一起使用上。在下例中，依据 IF 表达式的条件，可能不运行 DO 组，但总要运行其余的命令。

```

IF (&A=&B) THEN (DO)
    z
    z
    z
ENDDO

CHGVAR &C 8
SAVOBJzzz
CALL PGM (PAYROLL)
ENDPGM

```

仅当条件为真时运行

不管条件是否为真，无条件运行

如果 IF 的表达式不为真，用 ELSE 命令可以规定仅执一个命令或一组命令，则整个逻辑都改变了：

```

IF (&A=&B) THEN (DO)
    z
    z
    z
ENDDO

ELSE DO
    z
    z
    z
ENDDO

CHGVAR &C 8
SAVOBJ...
CALL PGM (PAYROLL)

```

仅在条件为真时做

仅在条件为假时做

无条件做

每个 ELSE 前必须有一个与之相关的 IF 命令，如果 IF 有嵌套，那么每个 IF 的 ELSE 都要与之匹配。

```

IF ... THEN ...
IF ... THEN(DO)
    IF ... THEN(DO)
        .
        .
    ENDDO
ELSE DO
    IF ... THEN(DO)
        .
        .

```



```
.  
    ENDDO  
ELSE DO  
.  
.  
.  
    ENDDO
```

```
    ENDDO  
ELSE IF ... THEN ...  
IF ... THEN ...  
IF ... THEN ...
```

要查过程中的ELSE 是否匹配，要从最里面的一组查起。

ELSE 也能用来检查一系列互相冲突的选项。在下例中，在成功地检查完第一个 IF 后，执行嵌套的命令和 RCLRSC 命令：

```
IF COND(&OPTION=1) THEN(CALLPRC PRC(ADDREC))  
ELSE  CMD(IF COND(&OPTION=2) THEN(CALLPRC PRC(DSPFILE)))  
ELSE  CMD(IF COND(&OPTION=3) THEN(CALLPRC PRC(PRINTFILE)))  
      ELSE CMD(IF COND(&OPTION=4) THEN(CALLPRC PRC(DUMP)))  
RCLRSC  
RETURN
```

#### 4.2.7.5 使用嵌套的 IF 命令

一个 IF 命令可以嵌入到另一个 IF 中，当在条件为真时，执行的是自身的另外一个 IF 命令（要用 THEN 参数）

```
IF (&A=&B) THEN(IF (&C=&D) THEN(GOTO END))  
GOTO START
```

在执行某个或某组命令前必须满足几个条件时，这种嵌套是很有用的。在前面的例子中，如果第一个表达式为真，系统读第一个 THEN 参数，假如&C=&D，表达式为真，系统执行第二个 THEN 的命令，即 GOTO END。二个表达式都为真，才能执行 GOTO END。如果有一个不为真，则执行 GOTO START。注意表达式和命令的括号。

在 CL 程序设计中可允许 10 层嵌套。

嵌套层越多，逻辑就变得越复杂。可以用自由格式使嵌套看起来更清楚：

```
PGM  
DCL &A *DEC 1  
DCL &B *CHAR 2  
DCL &RESP *DEC 1  
IF (&RESP=1) +  
    IF (&A=5) +
```

```
IF (&B=NO) THEN(DO)
```

```
.  
. .  
. .  
ENDDO
```

```
CHGVAR &A VALUE(8)
```

```
CALL PGM(DAILY)
```

```
ENDPGM
```

最前边的 IF 做一个嵌套命令处理，其中如有一个为假，则处理转移到嵌套外的语句（CHGVAR 及后续的命令），只有所有的为真，才执行 DO 组，这就比在一个命令中用\*AND 组合几个表达式要容易。

在某些情况下，有些转移必须是在条件为假时才发生，这时可以对每个嵌套的 IF 加一个 ELSE 语句：

```
PGM  
DCL &A ...  
DCL &B ...  
DCL &RESP ...  
IF (&RESP=1) +  
    IF (&A=5) +  
        IF (&B=NO) THEN(DO)  
            .  
            .  
            .  
            SNDPGMMMSG ...  
            .  
            .  
            .  
        ENDDO  
    ELSE CALLPRC PROCA  
ELSE CALLPRC PROCB  
CHGVAR &A 8  
CALLPRC PROC(DAILY)  
ENDPGM
```

如果所有条件为真，则执行SNDPGMMMSG，以及后边的CHGVAR，如果第一、第二个条件为真，第三个条件为假，则调用PROCA。当从PROCA返回时，执行CHGVAR。如果第二个条件为假，调用PROCB，接着执行CHGVAR，最后，如果&RESP≠1，立即执行CHGVAR命令。ELSE为每个检查提供不同的转移路径。

#### 4.2.7.6 使用\*AND、\*OR、和\*NOT 操作

\*AND 和\*OR 是逻辑操作的保留值，用来表示逻辑表达式操作之间的关系。\*AND 可以用&表示，\*OR 可以用|表示，后边必须有一空格。一个逻辑表达式的操作关系表达式或逻辑变量或常量用逻辑操作组合在一起。\*AND 表示两个操作都为真时才产生真的结果，\*OR 表示操作有一个为真就产生真的结果。其它非逻辑操作符，也可用在表达式中指出操作数之间的逻辑关系或表达式操作要完成的动作，除逻辑操作符外，有三类操作符：

- \_ 算术 (+, -, \*, /)
- \_ 字符 (\*CAT, ||, \*BCAT, |>, \*TCAT, |<)
- \_ 关系 (\*EQ, =, \*GT, >, \*LT, <, \*GE, >=, \*LE, <=, \*NE, ?=, \*NG, ?>, \*NL, ?<)

下面是逻辑表达式的例子：

```
((&C *LT 1) *AND (&TIME *GT 1430))
(&C *LT 1 *AND &TIME *GT 1430)
((&C < 1) & (&TIME>1430))
((&C< 1) & (&TIME>1430))
```

逻辑表达式由三部分组成：两个操作数和一个操作符，它是由操作符的类型（\*AND 和\*OR）来表示表达式是逻辑型的而不是由操作数类型确定。逻辑表达式中的操作数可以是逻辑变量或其它表达式。例如：

```
((&C *LT 1) *AND (&TIME *GT 1430))
```

整个逻辑关系放在括号中，操作数都是关系表达式，也分别由括号括起。从逻辑表达式中的第二个例子看出，操作数不非得放在括号中，但放在括号中就比较清晰，不用括号是因为\*AND 和\*OR 有不同的优先级。即先做\*AND，后做\*OR。对同一优先级的操作，要用括号控制操作顺序。

在命令中可以用关系表达式做条件：

```
IF (&A=&B) THEN(DO)
```

```
.
.
.
ENDDO
```

关系表达式中操作数可以是常量。

如果要规定多个条件，可以用逻辑表达式与关系表达式一起做操作数：

```
IF ((&A=&B) *AND (&C=&D)) THEN(DO)
```

```
.
.
.
ENDDO
```

由于逻辑表达式中也可用另外的逻辑表达式做操作数。那么可以形成复杂的逻辑关系：

```
IF (((&A=&B) *OR (&A=&C)) *AND ((&C=1) *OR (&D='0')))) THEN(DO)
```

此时，&D 是逻辑变量。

任何逻辑或关系表达式的结果都为‘0’或‘1’。如果整个表达式结果为1，则执行相关的命令，下面的命令解释了这种情况：

```
IF (((&A = &B) *AND (&C = &D)) THEN(DO)
      ((true'1') *AND (not true'0'))
      (not true '0'))
```

表达式最后结果为0，这样，不执行DO。

同样也可用逻辑变量来计算表达式：

```
PGM
DCL &A *LGL
DCL &B *LGL
IF (&A *OR &B) THEN(CALL PGM(PGMA))
.
.
.
ENDPGM
```

这时要计算条件表达式的值，看&A或&B是否有一个为1。如果是这样，整个表达式为真，则调用PGMA。

在逻辑变量或常量间用\*OR时，使用下列的真值表：

If &A is: '0' '0' '1' '1'

and &B is: '0' '1' '0' '1'

the OR expression is: '0' '1' '1' '1'

简单地说，对多个OR操作，如果所有值都为假，则结果为假，有一个为真则结果为真。

```
PGM
DCL &A *LGL VALUE('0')
DCL &B *LGL VALUE('1')
DCL &C *LGL VALUE('1')
IF (&A *OR &B *OR &C) THEN(CALL PGMA)
.
.
.
ENDPGM
```

此时值不全为假，则结果为真，调用PGMA。

在用\*AND和逻辑变量计算逻辑表达式时，用下面的真值表：

如果 &A 是: '0' '0' '1' '1'

且 &B 是: '0' '1' '0' '1'

AND的结果是: '0' '0' '0' '1'

对多个\*AND 操作, 有一个值为假则结果为假, 全为真时结果为真。

```
PGM
DCL &A *LGL VALUE('0')
DCL &B *LGL VALUE('1')
DCL &C *LGL VALUE('1')
IF (&A *AND &B *AND &C) THEN(CALL PGMA)
.
.
.
ENDPGM
```

此时值不全为真, 则结果为假, 不调用 PGMA。

在前面的这些例子中, 在操作数代表一个逻辑值时, 才能在表达式中使用逻辑操作数, 非逻辑型变量用 OR 或 AND 是不对的, 例如:

```
PGM
DCL &A *CHAR 3
DCL &B *CHAR 3
DCL &C *CHAR 3
```

不正确的是:

```
IF (&A *OR &B *OR &C = YES) THEN...
```

正确的是:

```
IF ((&A=YES) *OR (&B=YES) *OR (&C=YES)) THEN...
```

这时, 在关系操作符之间是 OR。

逻辑操作符\*NOT 用来取逻辑变量或常量的非。\*NOT 要在\*AND 或\*OR 前计算, \*NOT 后的操作要在任何操作数间逻辑关系前进行。

```
PGM
DCL &A *LGL '1'
DCL &B *LGL '0'
IF (&A *AND *NOT &B) THEN(CALL PGMA)
```

在此例中, 所有值都为真, 表达式才为真, 即调用 PGMA。

```
PGM
DCL &A *LGL
DCL &B *CHAR 3 VALUE('ABC')
DCL &C *CHAR 3 VALUE('XYZ')
```

```
CHGVAR &A VALUE(&B *EQ &C)
```

```
IF (&A) THEN(CALLPRC PROCA)
```

在这个例子中，值不为真，不调用 PROCA。

#### 4.2.8 系统资源的获取

##### 4.2.8.1 日期格式转换命令：CVTDAT

```
CVTDAT DATE (被转换日期) TOVAR (转换后日期) +  
FROMFMT (原格式) TOFMT (新格式) TOSEP (新分隔符)
```

其中：转换后日期的长度至少是：

- 对儒略日期(如 YMD, DMY 或 JUL 格式), 不使用分隔符为 5 个字符, 使用分隔符为 6 个字符。
- 对非儒略日期, 不使用分隔符为 6 个字符, 使用分隔符为 8 个字符。

日期格式和分隔符可以通过按 F4 键选择。

##### 4.2.8.2 检索系统值命令：RTVSYSVAL

```
RTVSYSVAL SYSVAL (系统值名) RTNVAR (CL 变量名)
```

其中：系统值名可以通过按 F4 键选择，如：系统当前日期的系统值是 QDATE。变量名必须和系统值的类型匹配，对字符型和逻辑型系统值而言，变量名的长度必须相符，对数字型，则长度不小于系统值长度。

##### 4.2.8.3 检索配置源命令：RTVCFGSRC

用于将现有配置的描述存放到源文件成员中

##### 4.2.8.4 配置状态检索命令：RTVCFGSTS

用于获取各类配置描述的现行状态，放入 CL 变量。

##### 4.2.8.5 检索网络属性的命令：RTVNETA

用于获取系统的网络属性，放入相应的 CL 变量。如：系统名 SYSNAME，本地网络标识 LCLNETID。

##### 4.2.8.6 检索作业属性命令：RTVJOBA

用于获取当前运行作业的属性，放入相应的 CL 变量。如：作业名 JOB，用户名 USER。

##### 4.2.8.7 检索对象描述命令：RTVOBJD

用于获取指定对象的描述属性，放入相应的 CL 变量。

##### 4.2.8.8 检索用户档案命令：RTVUSRPRF

用于获取指定用户的档案资料，放入相应的 CL 变量。

### 4.3 程序间的通讯

程序间的通讯是指不同程序之间数据或参数的传递和交流。这种通讯可以出现在不同种类的程序中。如：RPG 与 C 语言，也可以出现在不同机器上，如：PC 与 AS/400。针对 OS/400 而言，通常程序间的通讯具有三种方式：

CALL、RETURN 命令

数据队列通讯

数据域通讯

#### 4.3.1 CALL 命令的使用

调用程序：CALL PGM (PGMA) PARM (&A &B)

被调用程序：PGM PARM (&C &D)

有关 CALL 命令使用的几点说明：

1. 参数值可以是字符常量、数值常量、逻辑常量或 CL 常量，最多可达 40 个。
2. 参数值以 CALL 命令中出现的顺序传送，这必须与被调用程序的参数顺序相匹配，变量名不一定相同。
3. 被调程序中的接收参数必须说明，但接收值不受变量说明中初值的影响。
4. 接收参数值的改变会反映到调用程序中，但常量传送不会改变。
5. 字符常量通常以 32 个字节传送，数字常量以 15.5 长度压缩格式传送。

#### 4.3.2 数据队列的程序通讯

数据队列 (\*DTAQ) 是系统对象中的一种类型，当建立了这种对象后，一个程序可以发送数据给它，另一个程序再从中接收数据，从而达到程序之间的数据通讯。

##### 4.3.2.1 数据队列的优点

1. 数据队列是两个作业之间进行异步通讯的最快方法。相对数据库文件、消息队列或数据域而言，它需要较少的额外开销。
2. 多个作业可以向相同的数据队列送数据和取数据，而数据队列的先进先出、后进先出或关键字顺序排列属性，能够保证数据送取的正确性。
3. 在任何高级语言程序中，通过调用系统提供的程序，就可以对数据队列进行操作，而且操作方法灵活方便。数据队列的操作和使用包括两类：第一类使用 CL 命令；第二类调用系统程序。

CL 命令：

CRTDTAQ	建立数据队列
DLTDTAQ	删除数据队列
WRKDTAQ	工作数据队列

系统程序：

QSNDDTAQ	发送数据队列
QRCVDTAQ	接收数据队列
QCLRDTAQ	清除数据队列
QMHQRDQD	检索数据队列

#### 4.3.2.2 数据队列的发送

需要将数据发送给数据队列，只要在程序中调用 QSNDDTAQ。在 CL 程序中，调用的格式如下：

```
CALL PGM(QSNDDTAQ) PARM(&QNAME &LIB +  
                          &FLDLEN &FIELD &KEYLEN &KEY)
```

**&QNAME:** 是长度为 10 的字符型，它命名了数据队列，如：IN\_Q。**&LIB:** 是长度为 10 的字符型，它命名了数据队列所在的库，如：\*LIBL。

**&FLDLEN:** 是长度为 5 的数字型，它规定了发送给数据队列的字符数，如：100。

**&FIELD:** 是长度为&FLDLEN的字符型，它包含了具体发送给数据队列的数据。

**&KEYLEN:** 是长度为 3 的数字型，它说明了传送给数据队列的关键字长度，如：6。

**&KEY:** 是长度为&KEYLEN的字符型，它包含了传送给数据队列的关键字数据。注：后两个参数可以自选，如果说明了一个，则必须说明另一个。

#### 4.3.2.3 数据队列的接收

需要从数据队列中接收数据，只要在程序中调用 QRCVDTAQ。在 CL 程序中，调用的格式如下：

```
CALL PGM(QRCVDTAQ) PARM(&QNAME &LIB &FLDLEN +  
                          &FIELD &WAIT &ORDER KEYLEN &KEY &SNDRLLEN &SNDRL)
```

**&QNAME:** 是长度为 10 的字符型，它命名了数据队列。如：OUT\_Q。

**&LIB:** 是长度为 10 的字符型，它命名了数据队列所在的库。如：\*LIBL。

**&FLDLEN:** 是长度为 5 的数字型，它规定了发送给数据队列的字符数。

**&FIELD:** 是长度为&FLDLEN的字符型，它包含了从数据队列中接收到的具体数据。

**&WAIT:** 是长度为 5 的数字型，它说明了等待接收数据的时间。负数表示无限制的等待；零表示不等待；正数表示要等待的秒数，最大值是 9999。这个参数只有在数据队列中无满足条件的数据时，才起作用。

**&ORDER:** 是长度为 2 的字符型，它说明了按关键字接收数据的条件。可用的字符值是：GT、LT、EQ、GE、LE。

**&KEYLEN:** 是长度为 3 的数字型，它说明了接收数据队列的关键字长度。

**&KEY:** 是长度为&KEYLEN的字符型，它标识了用于从数据队列中接收数据的关键字变量。

**&SNDRLLEN:** 是长度为 3 的数字型，它规定了发送者标识的长度。

**&SNDRL:** 是长度为&SNDRLLEN的字符型，它包含了发送者标识的数据。

注：后三个参数可以任选，但是&ORDER、&KEYLEN和&KEY必须同时说明。

#### 4.3.2.4 数据队列的清除

需要从数据队列中清除数据，只要在程序中调用 QCLRDTAQ 在 CL 程序中，调用的格式如下：

```
CALL PGM (QCLRDTAQ) PARM (&QNAME &LIB)
```



#### 4.3.2.5 数据队列的检索

需要检索一个数据队列的描述项，只要在程序中调用 QMHQRDQD。在 CL 程序中，调用的格式如下：

```
CALL PGM (QMHQRDQD) PARM (&RCVR &RCVLEN +  
                             &FORMAT &DQNAME)
```

&RCVR: 是长度为&RCVLEN 的字符型，它标识了含有数据队列性的变量。

&RCVLEN: 是长度为 4 的数字型，它说明了&RCVR 长度。

&FORMAT: 是长度为 8 的字符型，它定义了接收模板的格式。

&DQNAME: 是长度为 20 的字符型，它标识了数据队列和所在库，前十个字符是队列名字，后十个字符是库名。

#### 4.3.3 数据域的程序通讯

数据域 (\*DTAARA) 是系统对象中的一种类型。当建立了这种对象后，可以用来存入数据，以便任何程序进行读取和修改。数据域的典型用途如下：

1. 提供用于几个程序中的常数字段，易于共享和修改。如：标题、说明等。
2. 在一个作业中提供一个传递信息的区域。
3. 在一个作业中提供一个字段作为控制参数，以便容易地得到修改。

数据域的 CL 命令包括：

CRTDTAARA	建立数据域，长度不超过 2000
CHGDTAARA	改变数据域，改变时数据域被锁定
DSPDTAARA	显示数据域，可以以十六进制方式显示
RTVDTAARA	检索数据域，检索值需要存入 CL 变量
DLTDTAARA	删除数据域
WRKDTAARA	工作数据域

### 4.4 测试功能

#### 4.4.1 CL 程序的编译

CL 源程序必须经过编译，生成 \*PGM 方可运行。建立程序的方法有两种：一是，使用 CL 命令 CRTCLPGM，二是，使用 PDM 菜单选项 14，按 F4 键即可对命令参数进行选择。

系统编译的情况全部记录在编译清单中，通过 WRKSPLF 即可看到。编译过程中的错误被列在相应命令后面，以及文件的最后，方便用户查找。下列类型的错误将停止程序的建立：

1. 值错
2. 句法错
3. 命令内部与参数间不符
4. 有效性检查有错

CL 程序可以通过反编译命令 RTVCLSRC，重新建立 CL 源程序。使用该命令时，必须满足建立编译程序的参数 ALWRTVSRC 为 \*YES。当源程序被反编后，任何注释信息不再重新产生，以下程序序言将被建立：

1. 所有者名
2. 源程序的最终修改日期
3. 源程序的重建日期和时间
4. 最初编译时的许可程序级

#### 4.4.2 CL 程序的测试

对于 CL 程序在编译和运行中的错误，系统提供以下几种测试功能：

1. 程序转储。在 CL 源程序中输入命令 DMPCLPGM，运行后通过 WRKSPLF 即可看到转储内容。包括程序信息队列的全部信息和全部变量的数值。如果程序运行出错，出现提示画面时，输入 D 也可进行程序转储。

2. 设置断点。第一步，启动测试环境，使用命令 STRDBG PGM (PGMA)；第二步，设置断点，使用命令 ADDBKP STMT (1500) PGMVAR( ' &A' ' &B' )；第三步，运行程序，显示断点信息；第四步，结束测试环境，使用命令 ENDBG。

3. 设置跟踪。跟踪是记录程序中语句执行顺序的过程。系统并不自动显示跟踪信息，而须使用命令 DSPTRCDTA 请求显示跟踪信息，信息包括语句执行的顺序和 ADDTRC 命令中指定的变量值。

## 第五章 显示文件

### 5.1 显示文件基本概念

显示文件是画面程序的画面装置部分，与其对应的是数据库文件或帐票文件部分，是实现用户与系统交互的一种方式。例如：程序需要把画面用户入力的条件接受，然后查询数据库，在把相关信息显示在画面上，而实现查询的功能。

通常情况，一个显示文件包括一个或多个记录，每个记录定义一个显示的所有特性，每个显示是由输出、输入、输入 / 输出字段和常量组成的。

在本章，我们将实例学习显示源文件的生成和显示文件创建。

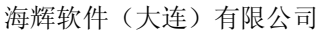
### 5.2 用 SEU 建立显示文件

使用源语句录入工具 SEU，通过 DDS 描述方式，创建 DSPF 类型的成员，然后编译生成 \*FILE 类型，DSPF 属性的对象，是创建显示文件的常用方法。在本节，我们先实例一个‘雇员信息一览’的画面显示文件，以此详细讲述显示文件创建过程，和其中的注意事项。

#### 5.2.1 显示文件程序范例

下面，是雇员信息一览的源程序：

```
A                                DSPSI Z(24 80 *DS3)
A                                CF03
A                                CF12
A      R FMT01
A                                OVERLAY
A      1 2' TEST01D'
A                                COLOR(BLU)
A      1 29' 雇员信息一览'
A                                DSPATR(RI)
A                                DSPATR(HI)
A      1 69SYSNAME
A                                COLOR(BLU)
A      2 2DATE
A                                EDTCDE(Y)
A                                COLOR(BLU)
A      2 69USER
A                                COLOR(BLU)
A      8 18' 雇员 I D:'
A                                COLOR(WHT)
A      D1SYCD      10A B 8 32TEXT(' 雇员 I D')
A                                DSPATR(CS)
A                                DSPATR(UL)
A      41
A      41                                DSPATR(RI)
A                                DSPATR(PC)
A      9 18' 雇员姓名:'
A                                DSPATR(HI)
A      D1SYNM      20 0 9 32TEXT(' 雇员名')
A                                12 18' 部門 I D:'
A                                COLOR(WHT)
A      D1BMCD      3 0 12 32TEXT(' 部門 I D')
A                                13 18' 职 位:'
A                                COLOR(WHT)
```



A	D1SYKY	3	0	13	32TEXT(' 职位')
A				15	18'入社日期:'
A					COLOR(WHT)
A	D1NYDT	8Y	00	15	32TEXT('入社日期')
A					EDTWRD(' / / ')
A	R FMT99				
A	D9CMD	780	0	23	2TEXT('CMD')
A	D9MSG	780	0	24	2TEXT('MSG')
A 99					DSPATR(RI)

### 编译后的显示文件(图中 TEST01D 的对象)

用 SDA 查看其描述的画面表示。

大连高新技术产业园区礼贤街 33 号                      - 76 -                      TEL: 0411-84556666  
FAX: 0411-84791350

### 5.2.2 显示文件程序编写

一般情况下，显示文件对应的源程序，都建立在专门放置数据库描述成员的文件中。为了管理，通常会把这个文件起名为 QDSPSRC. (有的项目也放在 QDDSSRC 中)我们也把社员信息一览 (TEST01D) 的描述文件，建立在 QDSPSRC 下。

首先，我们要启动源语句输入实用程序 (SEU)。在命令行键入 STRSEU，然后按 F4，配置参数，如下图：

Start Source Entry Utility (STRSEU)		
Type choices, press Enter.		
Source file . . . . .	<u>QDSPSRC</u>	Name, *PRV
Library . . . . .	<u>TST010</u>	Name, *LIBL, *CURLIB, *PRV
Source member . . . . .	<u>TEST010D</u>	Name, *PRV, *SELECT
Source type . . . . .	<u>DSPF</u>	Name, *SAME, BAS, BASP...
Option . . . . .	<u>2</u>	*BLANK, ' ', 2, 5, 6
Text 'description' . . . . .	<u>雇员信息一览</u>	
<hr/>		
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display		Bottom
F24=More keys		

#### 参数说明

源文件：建立位置说明之一。我们按默认规则，放在 QDSPSRC 文件下。

库：建立位置说明之一。我们放在 TST010 库下的 QDSPSRC 下。

源成员：为此文件的名称。

类型：我们建的是显示文件的源文件，所以，此处 DSPF

处理选择 (OPT)：键入 2 (编辑)

文件注释：键入文件说明信息，或文件名

配置好后，回车进入 SEU 编辑界面



因为，生成的类型中，我们键入的是 DSPF 类型，所以，进入编辑界面后，画面显示提示信息是 DP 的格式提示。在此处可以 SHIFT+F1 修改 SEU 默认显示模式，方便代码录入。

在这里，我们将逐行讲解实例代码的作用，和对应位置的其他规则。

Columns . . . . :	1	80	Edi t	TST010/QDSPSRC
SEU==>				TEST010D
FMT DP	.....	AAN01N02N03T.Name+++++RLen++TdpBLi nPosFuncti ons+++++	*****	*****
		*****	Beginni ngofdata	*****
0020.03	A		DSPSI Z(24 80 *DS3)	081013
0020.04	A		CF03	081013
0020.05	A		CF12	081013
0020.06	A	R FMT01		081013
0020.07	A		OVERLAY	081013
0020.08	A		1 2' TEST01D'	081013
0020.09	A		COLOR(BLU)	081013
0020.10	A		1 29' 雇员信息一览 '	081013
0020.11	A		DSPATR(RI)	081013
0020.12	A		DSPATR(HI)	081013
0020.13	A		1 69SYSNAME	081013
0020.14	A		COLOR(BLU)	081013
0020.15	A		2 2DATE	081013
0020.16	A		EDTCDE(Y)	081013
0020.17	A		COLOR(BLU)	081013
0020.18	A		2 69USER	081013
0020.19	A		COLOR(BLU)	081013
0020.20	A		8 18' 雇员 I D :	081013
0020.21	A		COLOR(WHT)	081013
0020.22	A	D1SYCD	10A B 8 32TEXT(' 雇员 I D')	081013
0020.23	A		DSPATR(CS)	081013
0020.24	A		DSPATR(UL)	081013
0020.25	A	41	DSPATR(RI)	081013
0020.26	A	41	DSPATR(PC)	081013
0020.27	A		9 18' 雇员姓名 :	081013
0020.28	A		DSPATR(HI)	081013
0020.29	A	D1SYNM	20A 0 9 32TEXT(' 雇员名 ')	081013
0020.30	A		12 18' 部 門 I D :	081013
0020.31	A		COLOR(WHT)	081013
0020.32	A	D1BMCD	3A 0 12 33TEXT(' 部 門 I D ')	081013
0020.33	A		13 18' 职 位 :	081013
0020.34	A		COLOR(WHT)	081013
0020.35	A	D1SYKY	3A 0 13 33TEXT(' 职 位 ')	081013
0020.36	A		15 18' 入社日期 :	081013
0020.37	A		COLOR(WHT)	081013
0020.38	A	D1NYDT	8Y 00 15 33TEXT(' 入社日期 ')	081013
0020.39	A		EDTWRD(' / / ')	081013
0020.40	A	R FMT99		081013
0020.41	A	D9CMD	780 0 23 2TEXT(' CMD')	081013
0020.41	A	D9CMD	780 0 23 2TEXT(' CMD')	081013
0020.42	A	D9MSG	780 0 24 2TEXT(' MSG')	081013
0020.43	A	99	DSPATR(RI)	081013

F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle  
F16=Repeat find F17=Repeat change F24=More keys

#### A1 列:

在提示行 FMT DP 标记的, ' A' 列, 在一下任何一行记录, 都要写 A, 此位置键入 A 来指定此表为 DDS 格式, 标记着这一行为有效记录。在 ' A' 后加\*, 表明此行为注释代码。

#### A2 列:

在提示行 FMT DP 标记的第 2 个 ' A' 列。有三种情况。' \*' : 表明此行为注释代码。' A' : 如果需多于三个指示器来构成一个 " 与 " 条件, 可在下一行或下几行中指定指示器。可在第二行或下几行位置的第 7 列键入 A 来继续 " 与 " 条件, 或此处为空白, 因为缺省值即为 A。' 0' : 如果指定的几个条件是 " 或 " 关系, 则每个条件必须各占一行, 且除第一个条件外, 第 7 列必须为 " 0"。为第一个条件指定 " 0" 会产生一条警告信息, 且此位置被假定为空白。

#### T 列:

类型项，此列有两种类型，R 说明定义了一个记录格式，如第 4 行和第 38 行的 T 列代码，紧跟其后边的是记录名，在描述字段时，此列不入力。H 说明帮助说明。

**N01N02N03 列：**

这里是三个指示器，控制后定义的功能或字段的条件限定。如果想把一个指示器置为 OFF，只需在此指示器之前位置指定“ N”，则表示 OFF 时为启用条件。如代码的第 41 行，当 99 为\*ON 时，后边的功能定义启用。

**姓名项：**

此列分两种情况，一种为记录格式名。如第 4 行代码，说明的是此表的记录格式名为 FMT01，一个显示文件可有多个记录格式名，但该文件中的每个记录格式名在此文件中都必须唯一。第二种为字段名。如 40 行的代码，此处的含义为字段的定义名。字段名不可以重复。最大长度为 10 位

**R 项：**

参照项：参照定义（R 表示只有当 T 列为空，NEME 列非空时有效，表示当前字段是一个参照字段，BLANK=当前字段非参照字段）

**LEN 项：**

长度项：字段长度。

**数据类型 T 项：**

数据类型 T 项：数据类型。常用的有效的数据类型有：

以下为显示文件的有效项：

项	含义	允许数据类型
空白	缺省	
X	仅为字母	字符
A	字母数值转换	字符
N	数值转换	字符或数值
S	有符号数值	数值
Y	仅为数值	数值
W	片假名（仅为日本用）	字符
I	禁止键盘输入	字符或数值
D	仅为数字	字符或数值
M	仅为数值字符	字符
F	浮点	数值

常用的类型为 S,Y,A。详细信息可以在代码处 F4, 把光标定位在数据类型处，按 F1 帮助查看。

**DP 列：**

小数点位数项：小数点位置定义，当数据类型 T 列定义为非数字型字段时，本栏应为空

**B 列：**

用这列来指定一个命名的字段是仅输出、仅输入、既输入又输出、隐藏、程序到系统或者信息字段。常量（未命名的）字段此位置无内容。显示文件的有效项如下：

项	含义
---	----



空格或 0	仅输出
I	仅输入
B	输入输出都可
H	隐藏（特殊的输出/输入字段）
M	信息（特殊的输出字段）
P	程序到系统（特殊的输出字段）

### LINPOS 列

使用这些列来为每个字段指定在显示器上的准确位置。不能为隐藏、程序到系统或信息字段指定位置。位置的有效与否是依据 **DSPSIZ** 键字和显示尺寸条件名。参见“**DSPSIZ**（显示尺寸）”的举例说明。

行（39 至 41 列）：用这些位置指定字段的起始行。该项必须为右对齐，前置零可选，最大行数为 24 或 27。

列（42 至 44 列）：使用这些位置来指定字段在确定行中的起始列位置，它为右对齐，前置零可选，一般最大列数为 80。（对 3180 设备为 132，）

功能定义，用于定义各种关键字

### FUNCTIONS 项：

本部分包括定义显示文件有效的键字项。这些键字在 45 至 80 列输入（功能项）。以下对常用键值进行讲解。

#### 1. DSPSIZ（显示尺寸）

用这个文件层键字定义用户程序所能打开的显示文件的显示尺寸。如示例程序第 1 行代码。

如果利用 IBM 提供的显示尺寸条件名。以任何顺序用 \*DS3 和 \*DS4 来定义，至少需要一个参数。但不能两次都定义同一个参数。

如果不用 IBM 提供的显示尺寸条件名，而是自己利用行和列位置来定义显示尺寸（只有 24×80，27×132 是有效的）。可以自己定义显示尺寸条件名而不使用 \*DS3 或 \*DS4。定义的显示尺寸条件名必须 2 到 8 个字符长，第一个字符必须是星号（\*）。要在字段层上的 DDS 语句的 7-16 列指定用户定义条件名。如果没指定用户定义条件名，则必须使用 IBM 提供的显示尺寸条件名。

可选指示器对这个键字无效。

#### 2. ALTPAGEDWN/ALTPAGEUP（替换的翻下页/翻上页键）

使用这两个文件层键字来把命令功能键(CF)定义为替换翻下页/翻上页键。当按 Page 键或 CF 键时，就调用页功能。选择翻下页/翻上页键只对文件中用 ALTPAGEDWN/ALTPAGEUP 键字定义的显示起作用，而对系统显示（比如帮助信息）不起作用。

这两个键字的格式为：

ALTPAGEDWN[（CFnn）]

ALTPAGEUP[（CFnn）]

可选参数的有效值为 CF01 到 CF24。如果没有指定参数，则 ALTPAGEDWN 缺省为 CF08，ALTPAGEUP 缺省为 CF07。

### 3. CFnn（命令功能）

使用此文件层或记录层键字来确定键 CF01 到 CF24 功能键。它和不传送修改了的数据的命令注意可以使用的键（CA）相反，这些功能键用作命令功能键（CF）来传送修改了的数据。响应指示器用 01 到 99。反映到程序当中。如实例代码的 2, 3 行。

### 4. OVERLAY

允许覆盖其他记录显示。既在显示本记录时，不清空已有显示记录。如本实例中，信息提示记录 FMT99，没有定义 OVERLAY，而上边的 FMT01 定义此键值。如代码第 5 行。写完信息提示记录后在写此记录，则可以实现画面上同时出现两条记录显示的效果。

### 5. ‘ ’ 文本输入

如代码的 6, 8。。。行。引号内的内容将显示在画面上。

### 6. DSPATR

字段显示属性的修饰功能键值，常用有效参数为：

值	含义
BL	闪烁
CS	列分隔符
HI	高亮度
NO	不显示
RI	反象显示
UL	下划线

### 7. COLOR

字段显示属性（颜色）的修饰功能键值，常用有效参数为：

值	含义
BLU	蓝色
GRN	绿色
PNK	粉红色
RED	红色
TRQ	蓝绿色
YLW	黄色
WHT	白色

如果没规定颜色参数，缺省值为白色。

### 8. EDTCDE（编辑码）

字段可用用户定义的编辑码做进一步的编辑。EDTCDE 可以满足大部分编辑要求。当它不能满足要求时，可使用 EDTWRD。OS/400 编辑码如下：1 至 4 ， A 至 D, J 至 Q, X 至 Z。

### 9. RTNCSRLOC（返回光标位置）

这是记录层键字，用来向应用程序返回光标位置，或光标所在字段名。

光标所在字段名：RTNCSRLOC(\*RECNAME &RCD &FLD &COL)

返回光标位置：RTNCSRLOC(\*WINDOW &LINNBR &POSNBR)

#### 10. SFL（子文件）

这是记录层键字，用来指定记录格式是子文件记录格式。记录格式（包括有关的字段描述）后面必须紧跟子文件控制记录格式（用 SFLCTL 键字指定）。

#### 11. SFLCLR（子文件清除）

这是记录层键字，用在子文件控制记录格式中，这样程序能清除所有记录的子文件。这个键字与 SFLDLT 不同，它不删除子文件，与 SFLINZ 也不同，它清除后子文件中不保存数据。清除子文件不会影响屏幕显示。然而，在清除后，子文件不保存活动记录。

此键字无参数。

#### 12. SFLCSRNRN（子文件光标相对记录号）

这是记录层键字，用在子文件控制记录格式中，返回子文件中光标所在处的相对记录号。如果子文件记录出现多行，此键字可与 SFLMODE 一起使用来确定光标位置。

键字格式为：SFLCSRNRN（&相对记录）

#### 13. SFLCTL（子文件控制）

这是记录层键字，用来指定记录格式是子文件控制记录格式。这个记录格式必须紧跟在子文件记录格式后面。

键字格式为：SFLCTL（子文件记录格式名）

下面是与 SFLCTL 一起使用的子文件键字的概括：（它们都是子文件控制记录中字段使用的字段层键字）。

子文件必须要定义的键值：SFLCTL，SFLDSP，SFLPAG，SFLSI Z

#### 14. SFLEND（子文件结束）

这是记录层键字，用在子文件控制记录格式中，允许在子文件或翻卷条所在显示的右下方显示加号（+）或标记（More...或 Bottom）加号（+）或标记 More...表示工作站用户可以通过按 PageUp 键来去掉标记，显示更多的记录。

下面给出画面显示文件有效的键字及对应的简要说明，供读者参考：

键字	功能简要说明	键字	功能简要说明
ALARM	报警	MLTCHCFLD	多选项选择字段
ALIAS	替换名	MNUBAR	菜单项
ALHELP	替换帮助键	MNUBARHC	菜单条选项
ALTNAME	替换记录名	MNUBARDSP	菜单条显示
ALTPAGEDWN	替换的翻下页	MNUBARSEP	菜单条分隔符
ALTPAGEUP	翻上页键	MNUBARSW	菜单条开关键
ALWGP	允许图形	MNUCNL	菜单取消键
ALWROL	允许滚动	MUUBTN	鼠标键
ASSUME	假定	MSGALARM	信息报警
AUTO	自动	MSGCON	信息常量
BLANKS	空白	MSGID	信息标识
BLINK	闪烁	MSGLOC	信息位置



BLKFOLD	空白折叠	NOCCSID	无编码字符集标识
Cann	命令注意	OPENPRT	打开打印机文件
CFnn	命令功能	OVERLAY	复盖
CHANGE	修改	OVRATR	复盖属性
CHCACCEL	选项加速说明	OVRDTA	复盖数据
CHCAVAIL	可用的选项颜色或显示属性	PAGEDOWN/PAGEUP	下页/上页键
CHCCTL	选项控制	PASSRCD	传送记录
CHCSLT	被选选项颜色/显示属性	PRINT	打印
CHCUNAVAIL	不可用选项的颜色/显示属性	PROTECT	保护
CHECK	检验	PSHBTNCHC	按钮字段选项
CHGINPDT	改变输入缺省值	PSHBTNFLD	按钮字段
CHKMSGID	检验信息标识	PULLDOWN	下拉菜单
CHOICE	选项	PUTOVR	放置明显复盖
CHRID	字符标识	PUTRETAIN	设置保留
CLEAR	清除	RANGE	范围
CLRL	清除行	REF	引用
CMP	比较	REFFLD	引用字段
CNTFLD	连续输入字段	RETCMDKEY	保持命令键
COLOR	颜色	RETKEY	保持功能键
COMP	比较	RETLCKSTS	保持加锁状态
CSRI NPNLY	光标移至仅输入位置	RMVWDW	取消窗口
CSRLOC	光标位置	RULLUP/RULLDOWN	翻/下翻
DATE	日期	RULLUP/RULLDOWN	例子
DFT	缺省	RTNCSRLOC	返回光标位置
DFTVAL	缺省值	RTNDTA	返回数据
DLTCHK	删除检验	SETOF	设置断开
DLTEDT	删除编辑	SETOFF	设置断开
DSPATR	显示属性	SFL	子文件
DSPMOD	显示方式	SFLCHCCTL	子文件选项控制
DSPRL	由右至左显示	SFLCLR	子文件清除
DSPSIZ	显示尺寸	SFLCSRPRG	子文件光标处理
DUP	重复	SFLCRRRN	子文件光标相对记录号
EDTCDE	编辑码	SFLCTL	子文件控制
DETMASK	编辑屏蔽	SFLDLT	子文件删除
EDTWRD	编辑字	SFLDROP	子文件撤消
ENTFLDATR	进入字段属性	SFLDSP	子文件显示
ERASE	清除	SFLDSPCTL	子文件显示控制



ERASEINP	清除输入	SFLEND	子文件结束
ERRMSG	错误信息	SFLENTER	子文件 enter 键
ERRMSGID	错误信息标识	SFLFOLD	子文件折叠
ERRSFL	错误子文件	SFLINZ	子文件初始化
FLDCSRPRG	光标前进字段	SFLLIN	子文件行
FLTFIXDEC	浮点到定点十进制	SFLMLTCHC	子文件多选项选择列表
FLTPCN	浮点精度	SFLMODE	子文件方式
FRCDTA	强制数据	SFLMSG	子文件信息
GETRETAIN	保留	SFLMSGID	子文件信息标识
HELP	帮助	SFLMSGKEY	子文件信息键
HLPARA	帮助区	SFLMSGRCD	子文件信息记录
HLPBDY	帮助边界	SFLNXTCHG	子文件的下一个修改
HLPCLR	帮助清除	SFLPAG	子文件页
HLPAMDKEY	帮助命令键	SFLPGMQ	子文件程序信息队列
HLPDOC	帮助文件	SFLRCDNBR	子文件记录号
HLPXCLD	禁止帮助	SFLRNA	子文件非活动记录
HLPFULL	全屏帮助	SFLROLYAL	子文件翻卷值
HLPID	帮助标识	SFLRTNSEL	子文件返回选择的选项
HLPPLNLRP	帮助面板组	SFLSCROLL	子文件上翻
HLPKCD	帮助记录	SFLSIZ	子文件大小
HLPRTN	帮助返回	SFLSNGCHC	子文件单选项选择列表
HLPSCIDX	帮助查询索引	SLNO	开始行号
HLPSEQ	帮助顺序	SNGCHCFLD	单项选项选择字段
HLPSELF	帮助书架	SYSNAME	系统名
HLPITLE	帮助标题	TEXT	正文
HTML	超级文本标识语言	TIME	时间
HOME	home 键	UNLOCK	解锁
INDARA	指示器区	USER	用户
INDTXT	指示器正文	USRDFN	用户定义
INVTTE	申请	USRDSPMGT	用户显示管理
INZINP	初始输入	USRRSTDSP	用户重存显示
INZRCD	初始记录	VALNUM	有效数字
KEEP	保持	VALUES	值
LOCK	加锁	VLDAMDKEY	有效命令键
LOGINP	日志输入	WDWBORDER	窗口边界
LOGOUT	日志输出	WDWTITLE	窗口标题
LOWER	小写	WINDOW	窗口

#### 5.2.4 编译生成对象文件

定义好显示源文件后，可以编译生成显示文件对象，程序才可以承载。用 CRTDSPF 命令创建显示文件。在 PDM 显示界面命令行，键入 CRTDSPF（也可在源物理文件前 OPT 键入 14），F4, 配置参数如下：

Create Display File (CRTDSPF)			
Type choices, press Enter.			
File . . . . .	> TEST010D	Name	
Library . . . . .	> TST010	Name, *CURLIB	
Source file . . . . .	> QDSPSRC	Name, *NONE	
Library . . . . .	> TST010	Name, *LIBL, *CURLIB	
Source member . . . . .	> TEST010D	Name, *FILE	
Generation severity level . . . .	20	0-30	
Flagging severity level . . . .	0	0-30	
Display device . . . . .	*REQUESTER	Name, *NONE, *REQUESTER	
+ for more values			
User specified DBCS data . . . .	*NO	*NO, *YES	
DBCS extension characters . . . .	*YES	*YES, *NO	
Text 'description' . . . . .	雇员信息一览		
More...			
F3=Exit	F4=Prompt	F5=Refresh	F10=Additional parameters
F13=How to use this display			F12=Cancel

如果，发生编译错误，可以 WRKJOB OPTION(\*SPLF)，查看编译清单提示信息，进行排除。

#### 5.2.5 显示子文件介绍

与简单的显示画面不同，可以显示子文件是可以同时对文件一组记录进行操作的显示文件界面。子文件通常由三个部分组成：

控制部分：显示附加信息，并且设置控制关键字

主体部分：数据库文件记录显示部分，显示一组记录

信息部分：信息显示部分，显示 F1、F2 等功能键和提示信息等内容

按照子文件的控制，分为全自动，半自动，和全手动三种。主要控制请参照 RPG 语言的讲解。

	全自动，半自动，全手动区别一览		
	全自动	半自动	全手动
写数据区别	全部写入	先写入一页，下翻页再写入	上下翻页都重新写入
优点	程序控制简单	速度比较快	速度快
缺点	速度与数据量成反比	介于全自动与手动之间	处理比较复杂

子文件的代码范例，请参照附件部分。

### 5.3 屏幕设计辅助工具（SDA）使用与说明

#### 5.3.1 SDA 简介

屏幕设计辅助工具（SDA）是 OS400 自带的系统工具。可以用来设计修改生成画面或菜单文件。当通过 DDS 生成显示文件对象后，也可以通过 SDA 显示该画面文件。在日常开发中，我们经常使用 DDS 定义显示源件，然后通过 SDA 显示画面，再在工作屏上对画面进行修改或变更，从而提高画面显示文件的创建效率。

SDA 与传统的设计显示的方法相比有以下优点：

1. DDS，用户不用非有 DDS 的有关知识。
2. 可用 SDA 来做给出一个功能组，很容易选择 DDS 的各层键字。允许从数据库文件中选择字段来设计显示。
3. 允许用数据以及所用指示器来测试显示。
4. 允许生成菜单和信息文件，SDA 用它来运行菜单。
5. 允许由 SDA 建立的 DDS 源语句生成显示文件。
6. 支持解释错误信息，在选择 DDS 键字时能判断出相冲突的源语句。

SDA 与 SEU 相同，也可处理 32764 行 DDS 源码。当用 SDA 生成修改或增加成员内容时，顺序号从 10 开始，增量为 10，可在规定其它选项的显示中修改顺序号及增量的值。

设计菜单时，每个菜单最多 148 个 DDS 记录，每个菜单映象有一个记录。菜单帮助最多 147 个记录，设计屏幕时，每个显示最多 149 个记录。

在本章，我们将实例创建一个与上一节 DDS 创建一样的显示画面，通过实例介绍 SDA 的基本功能和常用方法。

### 5.3.2 SDA 的启动

与其他 OS400 提供的工具一样，SDA 也有多种启动的方法。可以在命令行输入 STRSDA，按执行键，出现 SDA 主菜单。也可以从 PDM 启动 SDA，在 PDM 处理成员显示中，类型为 MNUDDS，MNUCMD 或 DSPF 的成员的 opt 列写 17，按执行键，根据所用成员的类型，出现设计屏幕显示或设计菜单显示。在这里，我们要用 SDA 实例创建一个 TEST01D 的简单显示画面，因为还没有成员，所以我们采用第一种方式启动。

在命令行键入 STRSDA 然后执行，进入屏幕设计辅助工具的主菜单。



AS/400 屏幕设计辅助工具 (SDA)

选择下列一项：

1. 设计屏幕

2. 设计菜单

3. 测试显示文件

选择或命令

====>

F1= 帮助    F3= 退出    F4= 提示    F9= 检索    F12= 取消

(C) COPYRIGHT IBM CORP. 1981, 2002.

5.3.3 通过 SDA 创建显示文件

在主菜单，因为我们要创建的是画面显示。所以，键入 1，然后回车。进入配置画面。

设计屏幕

输入选项，按“ 执行” 键。

源文件 . . . . .

库 . . . . .

成员 . . . . .

QDSPSRC

TST010

TEST010D

名称,    F4 显示列表

名称,    \*LIBL, \*CURLIB

名称,    F4 显示列表

F3= 退出    F4= 提示    F12= 取消

参数说明

源文件：存放成员的文件。一般按照惯例，放在 QDDSSRC，或 QDSPSRC 中。

库   ：文件所在的库。

成员：创建源文件的 ID。

然后执行





使用屏幕记录

文件 . . . . . : QDSPSRC  
库 . . . . . : TST010

成员 . . . . . : TEST010D  
源类型 . . . . . : DSPF

输入选项, 按“ 执行” 键。  
1= 添加                    2= 编辑注释                    3= 复制                    4= 删除  
7= 重新命名                8= 选择关键字                12= 设计图像

Opt    次序        记录                    类型                    相关子文件            日期                    DDS   错误

(文件中无记录)

F3= 退出                    F12= 取消                    F14= 文件级关键字  
F15= 文件级注释            F17= 子集                    F24= 其余键

底部

在这里, 我们要追加一个 FMT01 的记录。在 OPT 处填 1, 后边是记录名 FMT01 。执行

添加新记录

文件 . . . . . : QDSPSRC  
库 . . . . . : TST010

成员 . . . . . : TEST010D  
源类型 . . . . . : DSPF

输入选项, 按“ 执行” 键。

新记录 . . . . .                    FMT01                    名称

类型 . . . . .                    RECORD                    RECORD,    USRDFN  
   SFL,        SFLMSG  
   WINDOW,    WDWSFL  
   PULDOWN,   PDNSFL  
   MNUBAR

F3= 退出        F5= 刷新        F12= 取消

常用参数讲解:  
RECORD: 生成显示记录  
SFL: 为子文件记录选择记录层键字  
WINDOW: 生成窗口记录  
PULDOWN: 生成下拉记录  
MNUBAR: 生成菜单条的记录

画面项目设定

因为我们是简单的显示记录，因此我们默认选择 RECORD，执行进入 SDA 工作屏界面。在工作屏上，我们可以增加或修改常量和变量，也可以移动和删除他们。

#### 增加常量

在工作屏输入常量时，如果把字用单引号括起，生成包括所有字的常量。如果不用引号括起，则每个字生成一个常量。如，我们在设计屏键入 'TEST01D'，然后执行。则在对应的坐标位置上，生成了一个 TEST01D 的常量。

常量 \*DATE, \*TIME, \*USER, \*SYSNAME 是特别的输出常量，对它们要规定编辑值：

\*DATE: 6 位数字，缺省格式为 DD/DD/DD (月/日/年)

\*TIME: 8 位数字，格式为 TT: TT: TT (时: 分: 秒)

\*USER: 10 位字符，格式为 UUUUUUUUUU

\*SYSNAME: 8 位字符，格式为: SSSSSSSS

我们在画面上添加了如下图的常量。

```
TEST01D
DD/DD/DD
UUUUUUUUUU
SSSSSSSS
雇员信息一览
雇员 I D :
雇员姓名 :
部门 I D :
职 位 :
入社日期 :
```

#### 往工作屏加字段

在画面上，是可以直接引用数据库中的字段的方式添加字段的，但是因为代码规范等问题，现在已经不常用了。我们主要介绍直接加字段这种方式。

要往工作屏加字段：

1. 在要加字段的位置写加号(+)，它表示是属性字节位。
2. 在加号(+)后写下表中的字符，按执行键。这些字符决定了字段的类型：

字符	字段类型
3	数字输入字段
6	数字输出字段
9	数字输入/输出

字段

I 或 I 字符输入字段

O 或 o 字符输出字段

B 或 b 字符输入/输出

字段

M 或 m 信息常量字段

跟在字段定义后的可选圆括号表示长度或小数位，要如下那样定义：

1. 用+9(7,2)来定义一个长度为 7，小数位为 2 的即输入又输出的数字字段。
2. 用+iiii 来定义一个长度为 4 的输入字母数字字段。
3. +B(9)来定义一个长度为 9，输入又输出的字母数字字段。
4. 数字字段可以写小数点或逗号，仅 3、6、9 是有效的。所有其它数的缺省为 9。

在工作屏数字字段可定义为单精度(E)或双精度(D)的浮点格式，也能从数据库引用这些字段和放在工作屏，SDA 不允许在 16×64 显示尺寸上定义浮点字段。

可用大小写来规定数据类型，下表给出能定义的一些数字字段。

输入的	SDA 显示
+3(5,4)E	-3.3333E-333
+33.333d	-33.333D-333
+9(5,4)D	-9.9999D-999
+6(4,3)e	-6.666E-666

在字符字段中不允许有空格，因为空格用做字段界限。如果用的字符不是 I、O、B 或 M，那么字符的缺省值为 B。

用户定义字段的末尾空白位表示字段的结束。当按执行键，SDA 做：

在加号(+)之后校准字段。

根据字段的内容确定类型和长度。

对输出或输入/输出数字字段生成编辑码。

如果在第一位没给出用途字符，给出缺省值 9 或 B，分配一个从 FLD001 开始的字段名(FLDxxx)。如果以前已分配过，则此字段名为下个最高值。

下边是几个输入的举例：

输入	SDA 显示
+B(10)	<u>BBBBBBBBBB</u>
+0(20)	00000000000000000000
+0(3)	000
+0(3)	000

在空白位置上，我们添加一个字段，+B(10)，执行，显示为 BBBBBBBBBB。添加完字段后，按



照规范，我们要为这个字段命名（默认为 FLD001 然后连番）。在 BBBBBBBBBB 前键入？，然后执行。在下边会出现字段名（长度也可以修改），我们键入按照规范定义的字段 D1SYCD，执行。此字段就创建完了。

下面是我们实例的字段说明

字段名	数据长度及类型	I/O 类型
D1SYCD	10A	B
D1SYNM	20A	0
D1BMCD	3A	0
D1SYKY	3A	0
D1NYDT	8Y	0

```
TEST01D
DD/DD/DD
UUUUUUUUU
SSSSSSSS
雇员信息一览
雇员 I D :
雇员姓名 :      BBBBBBBBBB
部门 I D :      000000000000000000
职 位 :      000
入社日期 :      000
               66666666
```

#### 删除字段

在要删除的字段或常量前（属性位），键入 D，此字段就会被删除。

#### 复制字段

要复制一个字段或常量：先在要复制的字的属性位写一个减号，然后在要接收的属性位上写二个等号，执行即可。

如，我们要复制‘入社日期：’到指定位置（==处），则

```

TEST01D
DD/DD/DD
UUUUUUUUUU
SSSSSSSS
雇员信息一览
雇员 I D :
雇员姓名 :
部門 I D :
职 位 :
- 入社日期 :
BBBBBBBBBB
00000000000000000000
000
000
66666666

```

==

要复制多个字段：先在块的左上角写一个减号，在块的右下角写一个减号。然后在要接收的左上角写二个等号。执行即可。

当按执行键时，第一个减号的位置复制到两个等号位置上。二个减号之间的所有字符都被复制，完全或部分重叠的字段不复制。

字段居中

要把字段放在一行的中间，在字段的属性位上写 ac，按执行键。如果字段名以 c 开头，写 a 或 A 即可。按执行键就把此字段居中。

例如，我们要把‘雇员信息一览’居中，就可如下图：

```

TEST01D
DD/DD/DD
UUUUUUUUUU
SSSSSSSS
AC  雇员信息一览
雇员 I D :
雇员姓名 :
部門 I D :
职 位 :
入社日期 :
BBBBBBBBBB
00000000000000000000
000
000
66666666

```

执行后：

TEST01D  
DD/DD/DD  
UUUUUUUUUU  
SSSSSSSS

### 雇员信息一览

雇员 I D :  
雇员姓名 :  
部門 I D :  
职 位 :  
入社日期 :  
BBBBBBBBBB  
000000000000000000  
000  
000  
66666666

### 字段移动

要移的字段，在字段的属性位写减号(-)，在要移到的位置写等号(=)。也可以在要移动的字段属性位前（后）键入<(>)微调位置。在工作屏中不要用删除键和插入键移动字段或使字段变长变短。用这两个键会改变字段的起始位置，会引起不可预测的结果。

要移动多个字段则先在字段块的左上角写 -，在字段块的右下角写 -；然后在接收位置的左上角写 =，如下所示：

TEST01D  
DD/DD/DD  
UUUUUUUUUU  
SSSSSSSS

### 雇员信息一览

- 雇员 I D :  
雇员姓名 :  
部門 I D :  
职 位 :  
入社日期 :  
-  
BBBBBBBBBB  
000000000000000000  
000  
000  
66666666

=

执行，则

TEST01D DD/DD/DD UUUUUUUUUU SSSSSSSS	<p>雇员信息一览</p> <p>BBBBBBBBBB 00000000000000000000 000 000 66666666</p> <p>雇员 I D : 雇员姓名 : 部門 I D : 职 位 : 入社日期 :</p>
---	--

下面，我们把画面通过移动，设计成实例相同的布局。如下图：

TEST01D DD/DD/DD	雇员信息一览	SSSSSSSS UUUUUUUUUU
<p>雇员 I D :   BBBBBBBBBB 雇员姓名 :   00000000000000000000</p> <p>部門 I D :    000 职    位 :    000</p> <p>入社日期 :    66666666</p>		

下面，我们改变字段的显示属性，以改变字段的显示颜色，标识控制和显示格式与实例的画面显示属性相同。

规定颜色

在要改变显示属性的字段属性位键入\*, 如我们改‘雇员 I D : ’ 的显示属性为白色：

TEST01D DD/DD/DD	雇员信息一览	SSSSSSSS UUUUUUUUUU
<div style="margin-left: 100px;"> *雇员 I D :   BBBBBBBBBB  雇员姓名 :   00000000000000000000   <div style="margin-left: 40px;"> 部門 I D :     000  职     位 :     000   入社日期 :     66666666 </div> </div>		

执行进入字段属性定义画面，在颜色属性的选择上键入 Y 然后回车

选择字段关键字			
常量 . . . . . :	雇员 I D :	行 . . . . . :	列 . . . . . : 18
长度 . . . . . :	11		
输入选项，按“ 执行 ” 键。			
显示属性 . . . . . :	Y= 是	对字段类型 除“ 隐藏 ” 外的全部 除“ 隐藏 ” 外的全部	
颜色 . . . . . :	<u>Y</u>		
一般关键字 . . . . . _		所有类型	
TEXT 关键字 . . . . . _____			
F3= 退出     F4= 显示已选关键字     F12= 取消			

然后，在要显示的颜色上键入 1，如果要用标识控制颜色显示，可以把控制的标识写在后边。



选择颜色

常量 . . . . . : 雇员 I D :  
 长度 . . . . . : 11 行 . . . : 8 列 . . . : 18

输入选项，按“ 执行” 键。

颜色:	关键字	次序 (1-7)	指示符 /+
蓝 . . . . .	COLOR	—	— — —
绿 . . . . .	BLU	—	— — —
品红 . . . . .	GRN	—	— — —
红 . . . . .	PNK	—	— — —
青绿 . . . . .	RED	—	— — —
白 . . . . .	TRQ	—	— — —
黄 . . . . .	WHT	—	— — —
	YLW	—	— — —

F3= 退出      F12= 取消

两次回车执行之后，就可以看到如下

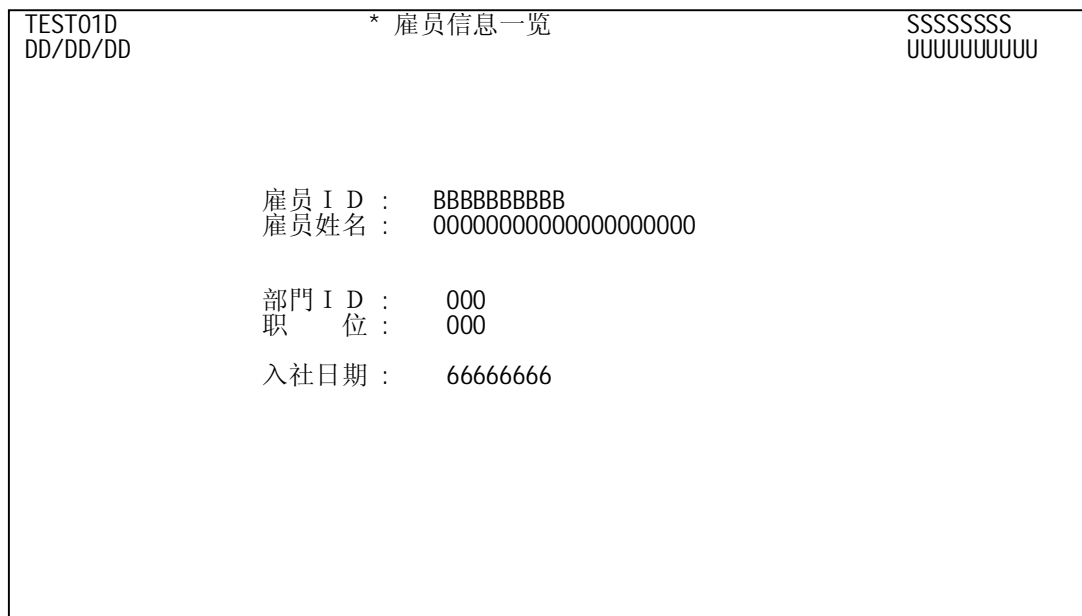


字段颜色变成了白色。我们可以按照相同的方法，把其他的字段都变成与范例对应相同的颜色。注意，最左边的字段，由于属性位不可入力，我们要先左移以为，属性定义好后在移回。



规定显示属性

按照实例，我们要把‘雇员信息一览’高亮反转。在要规定的字段属性位键入\*,如图：



回车，选择表示属性

选择字段关键字			
常量 . . . . . :	雇员信息一览	行 . . . . . :	1
长度 . . . . . :	14	列 . . . . . :	29
输入选项, 按“ 执行” 键。			
显示属性 . . . . . :	Y= 是	对字段类型	
颜色 . . . . . :	Y	除“ 隐藏” 外的全部	
	-	除“ 隐藏” 外的全部	
一般关键字 . . . . . :	-	所有类型	
TEXT 关键字 . . . . . : _____			
F3= 退出    F4= 显示已选关键字    F12= 取消			

回车执行

选择屏幕属性			
常量 . . . . . :	雇员信息一览	行 . . . . . :	1
长度 . . . . . :	14	列 . . . . . :	29
输入选项, 按“ 执行” 键。			
	关键字	Y= 是	标识符 /+
字段调节 . . . . . :			___ ___ ___
“ 程序至系统” 字段 . . . . . :			___ ___ ___
屏幕属性:	DSPATR		___ ___ ___
高亮度 . . . . . :	HI	Y	___ ___ ___
反转图像 . . . . . :	RI	Y	___ ___ ___
列分隔符 . . . . . :	CS	-	___ ___ ___
闪烁 . . . . . :	BL	-	___ ___ ___
非显示 . . . . . :	ND	-	___ ___ ___
下划线 . . . . . :	UL	-	___ ___ ___
字位光标 . . . . . :	PC	-	___ ___ ___
F3= 退出    F12= 取消			

高亮度处写 Y, 反转处写 Y, 两次回车后



同理，设置社员 ID 入力项为有分位下线，当标识 41 为\*ON 时，反转入力项，光标设定其上。  
如下图：

选择屏幕属性			
字段 . . . . .	D1SYCD	用法 . . .	B
长度 . . . . .	10	行 . . . .	8
		列 . . . .	32
输入选项，按“ 执行” 键。			
字段调节	关键字	Y= 是	标识符 /+
“ 程序至系统” 字段			— — —
屏幕属性:	DSPATR		— — —
高亮度	HI		— — —
反转图像	RI	Y	41 — — —
列分隔符	CS	Y	— — —
闪烁	BL	—	— — —
非显示	ND	—	— — —
下划线	UL	Y	— — —
字位光标	PC	Y	— — —
设置修改数据标记	MDT	—	— — —
保护字段	PR	—	— — —
操作员标识磁卡	OID	—	— — —
光笔选择	SP	—	— — —
F3= 退出 F12= 取消			

设置编辑格式

我们要是入社日期的显示项目的显示格式为 6666/66/66 的形式。在字段的属性位前入力\*, 执行，

选择编辑格式：

选择字段关键字			
字段 . . . . .	D1NYDT	用法 . . . . .	0
长度 . . . . .	8,0	行 . . . . .	15
		列 . . . . .	33
输入选项，按“ 执行” 键。			
显示属性 . . . . .	Y= 是	对字段类型	
颜色 . . . . .	—	除“ 隐藏” 外的全部	
一般关键字 . . . . .	—	所有类型	
编辑关键字 . . . . .	Y	数字输出或两者	
数据库引用 . . . . .	—	隐藏、输入、输出、两者	
错误消息 . . . . .	—	输入、输出、两者	
消息标识 (MSGID) . . . . .	—	输出或两者	
TEXT 关键字 . . . . .		入社日期	
F3= 退出      F4= 显示已选关键字      F12= 取消			

如图入力：



选择编辑关键字			
字段 . . . . .	D1NYDT	用法 . . . . .	0
长度 . . . . .	8,0	行 . . . . .	15
		列 . . . . .	33
输入选项，按“ 执行” 键。			
编辑代码 . . . . .	EDTCDE	关键字	尚有
替换前导 0，使用 . . . . .	-	A-D, J-Q, W, Y, Z, 1-9	
编辑字 . . . . .	EDTWRD	' / / '	
编辑屏蔽 . . . . .	EDTMSK		
F3= 退出      F12= 取消			

两次回车后



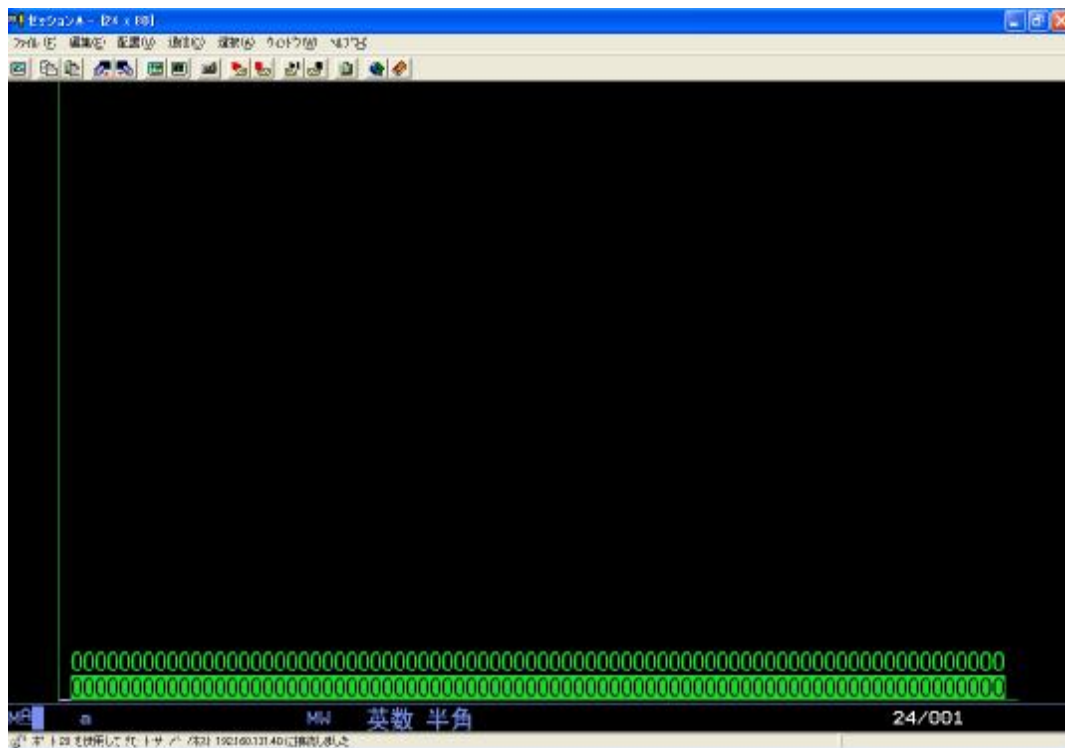
到此，我们实例的第一个记录就创建完成了。

#### 保存退出

要保存，并退出设计屏。只要键入 F3, 根据提示，选 1，即可退出当前记录设计，此时可以追加

其他记录。如果要退出 SDA, 则再次 F3, 确认源文件保存的路径和显示文件对象生成路径, 回车既可。

按照同样的方法, 我们还创建了 FMT99 记录。如下图



到此, 实例就创建完毕了, 本实例包含了 SDA 设计创建普通画面的最基本的常规用法。由于 DDS 描述创建显示文件的效率比较高, 而 SDA 的设计可见性比较明显, 因此, 实际项目中, 经常使用 DDS 的方式创建画面源文件, SDA 工具则经常用来作修改和变更字段的画面位置。所以 SDA 的高级使用方法, 这里就不详细介绍, 读者可以根据提示自己体会。

## 第六章 帐票文件（RLU）

### 6.1 基本概念

RLU 的全称为 Report Layout Utility(报表设计工具)。

RPG 程序设计的初衷就是为了设计报表以供系统打印。随着技术的改进和 OS/400 版本的不断升级，现在的许多输出设备都已经改变了传统报表的用法，原来使用程序定义定义文件设计报表，目前则经常采用报表设计工具 RLU 来设计打印文件 PRTF，然后再通过 RPGIV+PRTF 组合来编制程序。

### 6.2 举例说明 RLU 生成报表的过程

#### 6.2.1 PF 文件说明：

1、EDI SYSTEM/QDDSSRC(REFILE)为数据公共字典 PF：

A	R	REFMT	
A*	CHARACTER	SET	
A		CODE	5A
A		ID	7A
A*	DBCS-OPEN	SET	
A		NAME1	120
A		NAME2	120
A		DESCRPT	300
A**	NUMERIAL	SET	
A		AMOUNT	15 2
A		PRICE	11 2
A		QUATITY	7 0
A**	DATE	SET	
A		DATE1	L

2、EDI SYSTEM/QDDSSRC(EMPLOYEES)为参照数据公共字典的其中一个 PF：

A	*****		
A			REF(*LIBL/REFILE)
A			UNIQUE
A	R	EMPLOY	
A		EYCODE	R REFFLD(ID)
A			COLHDG('雇员码')
A		DPCODE	R REFFLD(CODE)
A			COLHDG('部门码')
A		EYNAME	R REFFLD(NAME2)
A			COLHDG('雇员名')
A		SUBTOTAL	R REFFLD(AMOUNT)
A			COLHDG('小计')
A*			
A		K EYCODE	
A		K DPCODE	



## 6.2.2 使用 RLU 生成报表的具体操作步骤:

### 6.2.2.1 为打印文件选择数据库:

1、在 OS/400 命令行敲入 STRRLU+F4，分别在 Source file、Library、Source member 输入对应内容，并且定义 Page width...198 。

```

                                Start Report Layout Utility (STRRLU)

Type choices, press Enter.
Source file . . . . . > QDDSSRC      Name, *PRV
Library . . . . . > XXXXXXXX      Name, *LIBL, *CURLIB, *PRV
Source member . . . . . > LPRTF01    Name, *PRV
Option . . . . . 2                2, 6
Page width . . . . . > 198          1-378, *SAME
Text 'description' . . . . . > 'DEMO for Design Report'

                                                                Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys

```

2、执行后出现如下效果，RLU 展示出报表设计屏面。

我们可以使用 SEU 对其进行编辑：左边的''''''为功能定义区域，空白区为设计编辑显示区域；左侧开始 BASE 显示了打印时出标尺；...+... 1 ...+... 2 最大可以定义 378 个字符位置。

```

Columns . . . :   1 71          Design Report          XXXXXX/XXXXXX
RLU==>;
LPRTF01
BASE   ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
----- Start of Page 001 -----
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
----- End of Report -----
F3=Exit   F11=Define field   F16=Delete field
F22=Alternative keys         F24=More keys

```



3、按 F10，输入 Library、File、Record，将光标停在 Field 栏按 F4：

Work	with	Database	Fields
Type options, press Enter. 1=Add field    4=Remove    8=Display field description			
Option	Field	Library	File
1		XXXXXXXX	XXXXXXXX
(No database fields selected)			
Bottom			

4、出现如下界面后再在 Field 栏按 F4:

Add Database Fields		
Type choices, press Enter.		
File . . . . .	XXXXXX	Name, F4 for list
ibrary . . . . .	XXXXXX	Name, *LIBL, *CURLIB
Record format . . . . .	XXXXX	Name, F4 for list
Field . . . . .		Name, F4 for list
F4=Prompt    F5=Refresh    F12=Cancel		

5、执行后可见下图，键入 1=Select，选择数据库所有的字段；然后连续键入 3 次执行：

Select Database Fields

File . . . . . : XXXXXX      Record . . . . . : XXXXX

Library . . . . . : XXXXXX

Position to . . . . .      Field

Subset . . . . . \*ALL      \*ALL, name, \*generic\*

Type options, press Enter.

1=Select    8=Display field description

Opt	Field	Length	Type	Column Heading
1	DPCODE	5	Character	DPCODE
1	EYCODE	7	Character	EYCODE
1	EYNAME	12	Bracketed DBCS	EYNAME
1	SUBTOTAL	15,2	Zoned decimal	SUBTOTAL

Bottom

F5=Refresh    F11=Display unsorted    F12=Cancel

6、我们可以发现刚才所选择的数据库字段已经放置在设计界面底端。

Columns . . . : 1 71      Design Report      XXXXXXXX/QDDSSRC

RLU==>      LPRTF01

BASE    ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7

----- Start of Page 001 -----

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

----- End of Report -----

1: DPCODE    2: EYCODE    3: EYNAME    4: SUBTOTAL

F3=Exit    F11=Define field    F16=Delete field

F22=Alternative keys      F24=More keys

### 6.2.2.2 定义记录格式:

打印过程是通过 RPGIV 程序使用写语句将记录输出写至打印文件。写语句后面所调用的是记录格式，每执行一次 RPGIV 写程序，就调用一次 RLU 文件中的记录格式，因此我们首先需要定义记录格式。

1、在左侧定义键入 DR(Define Record Format):

Columns . . . :	1 71	Design Report	XXXXXXXX/QDDSSRC
RLU==>;			LPRTF01
BASE	...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7		
	----- Start of Page 001 -----		
DR			
.....			
.....			
.....			
.....			
.....			
.....			
.....			
.....			
.....			
.....			
.....			
.....			
	----- End of Report -----		
1: DPCODE	2: EYCODE	3: EYNAME	4: SUBTOTAL
F3=Exit	F11=Define field	F16=Delete field	
F22=Alternative keys	F24=More keys		

2、键入执行后，可以看见 DR 定义自动生成的名为 RCD001 记录格式。

Columns . . . :	1 71	Design Report	XXXXXXXX/QDDSSRC
RLU==>;			LPRTF01
BASE	...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7		
	----- Start of Page 001 -----		
RCD001			
	----- End of Report -----		
1: DPCODE	2: EYCODE	3: EYNAME	4: SUBTOTAL
F3=Exit	F11=Define field	F16=Delete field	
F22=Alternative keys	F24=More keys		



```

Work with Record Keywords

Record format . . . . :   RCD001

Type options, press Enter.
    2=Specify    4=Remove

Opt      Keyword      Opt      Keyword      Opt      Keyword
CDEFNT   LPI
CHRSIZ   PAGRTT
CPI       PRTQTTY
DFNCHR   SKIPPA
DFNLIN   SKIPPB
DRAWER   SPACEA
FNTCHRSET SPACEB
FONT     TEXT
HIGHLIGHT
IGCCDEFNT
IGCCHRTT
INDTXT

Bottom

F3=Exit      F5=Refresh    F9=Input keyword parameters  F10=Rename record
F12=Cancel   F16=Remove all keywords

```

```

Rename Record Format

Number of keywords . . . . . :    0

Type choice, press Enter.

Record format . . . . .  DETAIL      Name

F3=Exit   F5=Refresh   F12=Cancel

```



```

Columns . . . :    1  71          Design Report          XXXXXXXX/QDDSSRC
RLU==>;                                     LPRTF01
BASE   ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
----- Start of Page 001 -----
DETAIL
----- End of Report -----

```

1、在 DETAIL 键入 VF(View Field)插入字段观察命令:



2、键入执行，可见左侧显示生成一个 FLD1 字段区准备放置 F10 所选择的数据库文件字段：

3、将编辑屏幕下方的数据库文件字段序号按字段比例键入对应位置，c 表示将字段表头放在上方。注意：一定要放置与 FLD1 同一行上；同时要预先估算一下留出的每个字段长度与下个字段长度

的间隔，防止字段相互覆盖。

Columns . . . :	1	71	Design Report	XXXXXXXX/QDDSSRC
RLU==>;				LPRTF01
BASE	...+... 1	...+... 2	...+... 3	...+... 4
	...+... 5	...+... 6	...+... 7	
	----- Start of Page 001 -----			
FLD1	1c	2c	3c	4c
DETAIL				
	----- End of Report -----			

1: DPCODE

2: EYCODE

3: EYNAME

4: SUBTOTAL

F3=Exit

F11=Define field

F16=Delete field

F22=Alternative keys

F24=More keys

4、键入执行，将数据库字段放置在设计报告上。

假设出现了字段间互相覆盖，可以使用 F13 消除字段区，F15 移动字段区，F16 字段删除，以调整各个字段间隔。

Columns . . . :	1	71	Design Report	XXXXXXXX/QDDSSRC
RLU==>;				LPRTF01
BASE	...+... 1	...+... 2	...+... 3	...+... 4
	...+... 5	...+... 6	...+... 7	
	----- Start of Page 001 -----			
FLD1	<...>;	<...>;	<...>;	<.....>;
RCD002	DPCODE	EYCODE	EYNAME	SUBTOTAL
FLD1	<...>;	<.....>;	<.....>;	<.....>;
DETAIL	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
	----- End of Report -----			
F3=Exit	F11=Define field	F16=Delete field		
F22=Alternative keys	F24=More keys			

5、在 DETAIL 命令行键入样本数据命令 SD5(Sample Data)，执行后参见下图，可见已经生成 5 行样本数据：





Columns . . . :	1	71	Design Report	XXXXXXXX/ODDSSRC
RLU==>;				LPRTF01
BASE	...+... 1	...+... 2	...+... 3	...+... 4
	...+... 5	...+... 6	...+... 7	
----- Start of Page 001 -----				
FLD1	<....>;	<....>;	<....>;	<.....>;
RCD002	DPCODE	EYCODE	EYNAME	SUBTOTAL
FLD1	<....>;	<.....>;	<.....>;	<.....>;
DETAIL	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00003 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00004 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00005 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00006 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00007 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
----- End of Report -----				
F3=Exit F11=Define field F16=Delete field				
F22=Alternative keys F24=More keys				

#### 6.2.2.4 定义表头:

1、在左侧命令行键入 I 插入一行，并键入 DR 定义一个记录格式，执行：

Columns . . . :	1	71	Design Report	XXXXXXXX/QDDSSRC
RLU==>				LPRTF01
BASE	...+... 1	...+... 2	...+... 3	...+... 4
	...+... 5	...+... 6	...+... 7	
----- Start of Page 001 -----				
DR				
FLD1	<....>;	<....>;	<....>;	<.....>;
RCD002	DPCODE	EYCODE	EYNAME	SUBTOTAL
FLD1	<...>;	<.....>;	<.....>;	<.....>;
DETAIL	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00003 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00004 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00005 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00006 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00007 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
----- End of Report -----				
F3=Exit F11=Define field F16=Delete field				
F22=Alternative keys F24=More keys				

2、然后在刚刚生成的记录格式上键入 DF，并且在同一编辑行居中位置键入“ hhhhhhhhhh”，定义一个常数字段：

Columns . . . :	1	71	Design Report	XXXXXXXX/QDDSSRC
RLU==>				LPRTF01
BASE	...+... 1	...+... 2	...+... 3	...+... 4
	...+... 5	...+... 6	...+... 7	
----- Start of Page 001 -----				
DF			hhhhhhhhhh	
FLD1	<....>;	<....>;	<....>;	<.....>;
RCD002	DPCODE	EYCODE	EYNAME	SUBTOTAL
FLD1	<...>;	<.....>;	<.....>;	<.....>;
DETAIL	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00004 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00005 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00006 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00007 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00008 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
----- End of Report -----				
F3=Exit F11=Define field F16=Delete field				
F22=Alternative keys F24=More keys				

### 3、键入执行，生成新的记录格式名 RCD004 以及表头字段：

```

Columns . . . :   1  71          Design Report          XXXXXXXX/QDDSSRC
RLU==>;
BASE   ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
----- Start of Page 001 -----
FLD1
RCD004          <.....>;
                hhhhhhhhhh
FLD1   <...>;   <...>;   <...>;   <...>;
RCD002  DPCODE  EYCODE          EYNAME          SUBTOTAL
FLD1   <...>;   <...>;   <.....>;   <.....>;
DETAIL  XXXXX   XXXXXXXX   XXXXXXXXXXXXX   999999999999999
00004 S  XXXXX   XXXXXXXX   XXXXXXXXXXXXX   999999999999999
00005 S  XXXXX   XXXXXXXX   XXXXXXXXXXXXX   999999999999999
00006 S  XXXXX   XXXXXXXX   XXXXXXXXXXXXX   999999999999999
00007 S  XXXXX   XXXXXXXX   XXXXXXXXXXXXX   999999999999999
00008 S  XXXXX   XXXXXXXX   XXXXXXXXXXXXX   999999999999999
----- End of Report -----

F3=Exit   F11=Define field   F16=Delete field
F22=Alternative keys   F24=More keys

```

### 4、使用 F18 然后再 F10，跟上面同样方法改变 RCD004 记录格式名为：HEADER

```

Columns . . . :   1  71          Design Report          XXXXXXXX/QDDSSRC
RLU==>;
BASE   ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
----- Start of Page 001 -----
FLD1
HEADER          <.....>;
                hhhhhhhhhh
FLD1   <...>;   <...>;   <...>;   <...>;
RCD002  DPCODE  EYCODE          EYNAME          SUBTOTAL
FLD1   <...>;   <...>;   <.....>;   <.....>;
DETAIL  XXXXX   XXXXXXXX   XXXXXXXXXXXXX   999999999999999
00004 S  XXXXX   XXXXXXXX   XXXXXXXXXXXXX   999999999999999
00005 S  XXXXX   XXXXXXXX   XXXXXXXXXXXXX   999999999999999
00006 S  XXXXX   XXXXXXXX   XXXXXXXXXXXXX   999999999999999
00007 S  XXXXX   XXXXXXXX   XXXXXXXXXXXXX   999999999999999
00008 S  XXXXX   XXXXXXXX   XXXXXXXXXXXXX   999999999999999
----- End of Report -----

F3=Exit   F11=Define field   F16=Delete field
F22=Alternative keys   F24=More keys

```

5、将光标移至常数字段 hhhhhhhhhh 上，键入 F23 进行编辑字段：

Work with Field Keywords					
Field . . . . . :		FLD001		Record format . . . . . : HEADER	
Type options, press Enter. 2=Specify 4=Remove					
Opt	Keyword	Opt	Keyword	Opt	Keyword
	ALIAS		DFT		IGCCHRTT
	BARCODE		DLT EDT		INDTXT
	BLKFOLD		EDTCDE		MSGCON
	CDEFNT		EDTWRD		PAGNBR
	CHRID		FLT FIXDEC		PRTQ LTY
	CHRSIZ		FLTPCN		REFFLD
	COLOR		FNTCHRSET		SKI PA
	CPI		FONT		SKI PB
	CVTDTA		HIGHLIGHT		SPACEA
	DATE		IGCALT TYP		SPACEB
	DATFMT		IGCANKCNV		TEXT
	DATSEP		IGCDEFNT		TIME
					More...
F3=Exit F5=Refresh F9=Input keyword parameters F10=Specify information F12=Cancel F16=Remove all keywords					

6、在 DFT 选项上键入 2 编辑常数项，执行，键入 ' 职工信息 '，此处定义的是本报表的表头，然后再键入 2 次执行：

Specify Default	
Keyword . . . . . :	DFT
Field . . . . . :	FLD001
Record format . . . . . :	HEADER
Type text of constant, press Enter.	
' 职工信息 '	
F3=Exit F5=Refresh F12=Cancel F16=Remove keyword	

7、键入 F10，然后再键入 F11 将其转换为常数项：

Specify Field Information		
Edited length . . . . .	:	10
Record format . . . . .	:	HEADER
Number of keywords . . . . .	:	1
Number of indicators . . . . .	:	0
Constant keyword . . . . .	:	' 职工信息 '
Type choices, press Enter.		
Option indicators . . . . .		01-99, N01-N99
More indicators . . . . .	N	Y=Yes, N=No
Starting line . . . . .		1-255
Starting position . . . . .	23	1-255, +nn
F3=Exit F5=Refresh F11=Convert to named field F12=Cancel		

8、按 2 次执行键退出编辑。

用 Work with Field Keywords 同样的定义方法，可以对表头字体进行编辑，下图显示了定义字形 CHRSIZ 为 2×2 时的编辑状态。

注意：只有当打印以后才能看见实际字形，而在 5250 终端上显示仅仅是普通尺寸字体。

Specify Character Size		
Keyword . . . . .	:	CHRSIZ
Field . . . . .	:	FLD005
Record format . . . . .	:	HEADER
Type choices, press Enter.		
Expand character size:		
Width multiplier . . . . .	2	1.0-20.0
Height multiplier . . . . .	2	1.0-20.0
F3=Exit F5=Refresh F12=Cancel F16=Remove keyword		

# 9、同时定义表头跳页关键字 SKIPB:

Work with Field Keywords					
Field . . . . . :		FLD005		Record format . . . . . : HEADER	
Type options, press Enter. 2=Specify 4=Remove					
Opt	Keyword	Opt	Keyword	Opt	Keyword
	ALIAS		DFT		IGCCHRRTT
	BARCODE		DLTEDT		INDTXT
	BLKFOLD		EDTCDE		MSGCON
	CDEFNT		EDTWRD		PAGNBR
	CHRID		FLTFIXDEC		PRTQLTY
>	CHRSIZ		FLTPCN		REFFLD
	COLOR		FNTCHRSET		SKIPB
	CPI		FONT	2	SKIPB
	CVTDTA		HIGHLIGHT		SPACEA
	DATE		IGCALTYP		SPACEB
	DATFMT		IGCANKCNV		TEXT
	DATSEP		IGCCDEFNT		TIME
More...					
F3=Exit F5=Refresh F9=Input keyword parameters F10=Specify information					
F12=Cancel F16=Remove all keywords					

10、为什么要定义 SKIPB 呢？对于链式打印纸来说，每次换页时，应该跳过纸缝再打印表头，SKIPB 就是完成这个功能的。键入执行，完成对表头的关键字的设计。在表头设计我们共定义了 CHRSIZ、SKIPB、DFT 这 3 个关键字。

Specify Skip Before		
Keyword . . . . . :	SKIPB	
Field . . . . . :	FLD005	
Record format . . . . . :	HEADER	
Number of indicators . . . . . :	0	
Type choices, press Enter.		
Line number to skip to before printing . . . . .	1	1-255
Option indicators . . . . .		01-99, N01-N99
More indicators . . . . .	N	Y=Yes, N=No
F3=Exit F5=Refresh F12=Cancel F16=Remove keyword		

## 11、键入执行返回设计报告：

Columns . . . :	1	71	Design Report	XXXXXXXX/QDDSSRC
RLU==>;				LPRTF01
BASE	...+... 1	...+... 2	...+... 3	...+... 4
	...+... 5	...+... 6	...+... 7	
----- Start of Page 001 -----				
FLD1	<.....>;			
HEADER	职工信息			
FLD1	<.....>;	<.....>;	<.....>;	<.....>;
RCD002	雇员码	部门码	雇员名	小计
FLD1	<...>;	<.....>;	<.....>;	<.....>;
DETAIL	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00004 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00005 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00006 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00007 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00008 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
----- End of Report -----				
F3=Exit F11=Define field F16=Delete field				
F22=Alternative keys F24=More keys				

## 12、定义表头页号，使得每打印 1 页，表头页号自动加 1。将光标移动至表头的右上侧，键入 F11 定义一个页号字段：

Define Field Information			
Edited	length		1
Record format . . . . .	:	HEADER	
Number of keywords . . . . .	:	0	
Number of indicators . . . . .	:	0	
Type choices, press Enter.			
Field . . . . .	FLD009	Name	
Option indicators . . . . .		01-99, N01-N99	
More indicators . . . . .	N	Y=Yes, N=No	
Starting line . . . . .		1-255	
Starting position . . . . .	+31	1-255, +nn	
Length of data . . . . .	1	1-378, +nn, -nn	
More...			
F3=Exit F5=Refresh F10=Work with keywords F11=Convert to constant field			
F12=Cancel			

13、再键入 F10，在关键字 PAGNBR 选项键入 2：

Work with Field Keywords					
Field . . . . . :		FLD009		Record format . . . . . : HEADER	
Type options, press Enter. 2=Specify 4=Remove					
Opt	Keyword	Opt	Keyword	Opt	Keyword
	ALIAS		DFT		IGCCHRTT
	BARCODE		DLTDT		INDTXT
	BLKFOLD		EDTCDE		MSGCON
	CDEFNT		EDTWRD	2	PAGNBR
	CHRID		FLTIXDEC		PRTQTTY
	CHRSIZ		FLTPCN		REFFLD
	COLOR		FNTCHRSET		SKIP
	CPI		FONT		SKIPB
	CVTDTA		HIGHLIGHT		SPACEA
	DATE		IGCALTYP		SPACEB
	DATFMT		IGCANKCNV		TEXT
	DATSEP		IGCDEFNT		TIME
					More...
F3=Exit F5=Refresh F9=Input keyword parameters F12=Cancel F16=Remove all keywords					

14、键入 4 次执行，定义页号关键字 PAGNBR，在定义过程中按执行不必理睬在提示行出现的 ERROR 信息；返回到 Specify Field Information 界面：

Specify Field Information		
Edited length . . . . . :	1	
Record format . . . . . :	HEADER	
Number of keywords . . . . . :	1	
Number of indicators . . . . . :	0	
Type choices, press Enter.		
Field . . . . .	FLD009	Name
Option indicators . . . . .		01-99, N01-N99
More indicators . . . . .	N	Y=Yes, N=No
Starting line . . . . .		1-255
Starting position . . . . .	+31	1-255, +nn
Length of data . . . . .	1	1-378, +nn, -nn
		More...
F3=Exit F5=Refresh F10=Work with keywords F11=Convert to constant field F12=Cancel		



15、键入 F11 将其转换为常数字段。键入执行，可见 PAGNBR 字段定义在表头右侧：

```

Columns . . . :   1  71          Design Report          XXXXXXXX/QDDSSRC
RLU==>;                                LPRTF01
BASE   ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
----- Start of Page 001 -----
FLD1          <.....>;                                <...>;
HEADER          职工信息                                9999
FLD1          <.....>;          <.....>;          <.....>;
<.....>;
RCD002      雇员码          部门码          雇员名          小计
FLD1      <...>;          <.....>;          <.....>;          <.....>;
DETAIL      XXXXX          XXXXXXXX          XXXXXXXXXXXXX          999999999999999
00004 S      XXXXX          XXXXXXXX          XXXXXXXXXXXXX          999999999999999
00005 S      XXXXX          XXXXXXXX          XXXXXXXXXXXXX          999999999999999
00006 S      XXXXX          XXXXXXXX          XXXXXXXXXXXXX          999999999999999
00007 S      XXXXX          XXXXXXXX          XXXXXXXXXXXXX          999999999999999
00008 S      XXXXX          XXXXXXXX          XXXXXXXXXXXXX          999999999999999
----- End of Report -----

F3=Exit   F11=Define field   F16=Delete field
F22=Alternative keys       F24=More keys

```

16、在左侧命令行上 HEADER 记录格式上键入 DF (DeFiNe FiEl d)，同时在 PAGNBR 字段左侧相应位置键入 Page:，执行后在页号字段左侧生成常数字段 Page:

```

Columns . . . :   1  71          Design Report          XXXXXXXX/QDDSSRC
RLU==>;                                LPRTF01
BASE   ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
----- Start of Page 001 -----
FLD1          <.....>;                                <...>;
DF          职工信息                                Page: 9999
FLD1      <.....>;          <.....>;          <.....>;          <.....>;
RCD002      雇员码          部门码          雇员名          小计
FLD1      <...>;          <.....>;          <.....>;          <.....>;
DETAIL      XXXXX          XXXXXXXX          XXXXXXXXXXXXX          999999999999999
00004 S      XXXXX          XXXXXXXX          XXXXXXXXXXXXX          999999999999999
00005 S      XXXXX          XXXXXXXX          XXXXXXXXXXXXX          999999999999999
00006 S      XXXXX          XXXXXXXX          XXXXXXXXXXXXX          999999999999999
00007 S      XXXXX          XXXXXXXX          XXXXXXXXXXXXX          999999999999999
00008 S      XXXXX          XXXXXXXX          XXXXXXXXXXXXX          999999999999999
----- End of Report -----

F3=Exit   F11=Define field   F16=Delete field
F22=Alternative keys       F24=More keys

```

17、键入执行，现在报表设计完成如下：

Columns . . . :	1	71	Design Report	XXXXXXXX/QDDSSRC
RLU==>;				LPRTF01
BASE	...+... 1	...+... 2	...+... 3	...+... 4
	...+... 5	...+... 6	...+... 7	
----- Start of Page 001 -----				
FLD1	<.....>;			<...>;<...>;
HEADER	职工信息			Page: 9999
FLD1	<.....>;	<.....>;	<.....>;	<.....>;
RCD002	雇员码	部门码	雇员名	小计
FLD1	<...>;	<.....>;	<.....>;	<.....>;
DETAIL	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00004 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00005 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00006 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00007 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00008 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
----- End of Report -----				
F3=Exit F11=Define field F16=Delete field				
F22=Alternative keys F24=More keys				

#### 6.2.2.5 合并记录格式

我们见到记录格式一共有三个：HEADER、DETAIL、RCD002，需要将 HEADER 与 RCD002 合并在一起。

1、在左侧命令行 RCD002 上键入 CLC(Change Line Format and Combine):

Columns . . . :	1	71	Design Report	XXXXXXXX/QDDSSRC
RLU==>;				LPRTF01
BASE	...+... 1	...+... 2	...+... 3	...+... 4
	...+... 5	...+... 6	...+... 7	
----- Start of Page 001 -----				
FLD1	<.....>;			<...>;<...>;
HEADER	职工信息			Page: 9999
FLD1	<.....>;	<.....>;	<.....>;	<.....>;
RCD002	雇员码	部门码	雇员名	小计
FLD1	<...>;	<.....>;	<.....>;	<.....>;
DETAIL	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00004 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00005 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00006 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00007 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00008 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
----- End of Report -----				
F3=Exit F11=Define field F16=Delete field				
F22=Alternative keys F24=More keys				

2、键入执行,可见 RCD002 被合并到 HEADER 记录格式中,在其行最左侧显示+表示此行为 HEADER 的记录格式续行:

Columns . . . :	1	71	Design Report	XXXXXXXX/ODDSSRC
RLU==>;				LPRTF01
BASE	...+... 1	...+... 2	...+... 3	...+... 4
	...+... 5	...+... 6	...+... 7	
----- Start of Page 001 -----				
FLD1	<.....>;			<...>;<...>;
HEADER	职工信息			Page: 9999
FLD1	<.....>;	<.....>;	<.....>;	<.....>;
00002 +	雇员码	部门码	雇员名	小计
FLD1	<...>;	<...>;	<.....>;	<.....>;
DETAIL	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00004 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00005 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00006 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00007 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00008 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
----- End of Report -----				
F3=Exit F11=Define field F16=Delete field				
F22=Alternative keys F24=More keys				

3、进一步使用 DR（定义一个记录格式）、DF（定义虚线字段）、CLC（合并记录格式至表头）生成按字段宽度分割虚线,即每一个字段与标题之间生成'-----' 修饰报表。

Columns . . . :	1	71	Design Report	XXXXXXXX/ODDSSRC
RLU==>;				LPRTF01
BASE	...+... 1	...+... 2	...+... 3	...+... 4
	...+... 5	...+... 6	...+... 7	
----- Start of Page 001 -----				
FLD1	<.....>;			<...>;<...>;
HEADER	职工信息			Page: 9999
FLD1	<.....>;	<.....>;	<.....>;	<.....>;
<.....>;				
00002 +	雇员码	部门码	雇员名	小计
FLD1	<.....>;	<.....>;	<.....>;	<.....>;
00003 +	-----	-----	-----	-----
FLD1	<...>;	<...>;	<.....>;	<.....>;
DETAIL	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00005 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00006 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00007 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00008 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
00009 S	XXXXX	XXXXXXX	XXXXXXXXXXXX	99999999999999
----- End of Report -----				
F3=Exit F11=Define field F16=Delete field				
F22=Alternative keys F24=More keys				

#### 6.2.2.6 建立变量型字段

1、首先使用 DR、DF 以及重命名功能定义一个记录格式 TOTALN。将光标移至小计字段 SUBTOTAL 下方，键入 F11 定义一个汇总字段。 字段名我们定义为 TOTALX，字段长度 17，数据类型为 2。

Define Field Information		
Edited length . . . . .	:	1
Record format . . . . .	:	TOTALN
Number of keywords . . . . .	:	0
Number of indicators . . . . .	:	0
Type choices, press Enter.		
Field . . . . .	TOTALX	Name
Option indicators . . . . .		01-99, N01-N99
More indicators . . . . .	N	Y=Yes, N=No
Starting line . . . . .		1-255
Starting position . . . . .	51	1-255, +nn
Length of data . . . . .	17	1-378, +nn, -nn
More...		
F3=Exit F5=Refresh F10=Work with keywords F11=Convert to constant field		
F12=Cancel		

2、PageDown 下翻一页，定义小数点位置为 2 位，参照字段 Y，使用参照值为 N:

Define Field Information		
Edited length . . . . .	:	1
Record format . . . . .	:	TOTALN
Number of keywords . . . . .	:	0
Number of indicators . . . . .	:	0
Type choices, press Enter.		
Data type . . . . .	2	1=Character 2=Zoned 3=Floating point 4=Open 5=Graphic 6=Date 7=Time 8=Time stamp
Decimal positions . . . . .	2	0-31, +n, -n
Reference a field . . . . .	Y	Y=Yes, N=No
Use referenced values . . . . .	N	Y=Yes, N=No
Bottom		
F3=Exit F5=Refresh F10=Work with keywords F11=Convert to constant field		
F12=Cancel		

3、按 2 次 F10 进入 Work with Field Keywords, 在 REFFLD 使用选项 2 定义参照数据库字段:

Work with Field Keywords					
Field . . . . . :		TOTALX		Record format . . . . . : TOTALN	
Type options, press Enter. 2=Specify 4=Remove					
Opt	Keyword	Opt	Keyword	Opt	Keyword
	ALIAS		DFT		IGCCHRTT
	BARCODE		DLTDT		INDTXT
	BLKFOLD		EDTCDE		MSGCON
	CDEFNT		EDTWRD		PAGNBR
	CHRID		FLTFIXDEC		PRTQTY
	CHRSIZ		FLTPCN	2	REFFLD
	COLOR		FNTCHRSET		SKIPPA
	CPI		FONT		SKIPPB
	CVTDTA		HIGHLIGHT		SPACEA
	DATE		IGCALTYP		SPACEB
	DATFMT		IGCANKCNV		TEXT
	DATSEP		IGCCDEFNT		TIME
					More...
F3=Exit F5=Refresh F9=Input keyword parameters F12=Cancel F16=Remove all keywords					

4、键入执行, 在 Field 使用 F4, 选择对应字段 SUBTOTAL, 然后再键入 3 次执行返回设计报告:

Specify Referenced Field		
Keyword . . . . . :	REFFLD	
Field . . . . . :	TOTALX	
Record format . . . . . :	TOTALN	
Type choices, press Enter.		
Field . . . . .	SUBTOTAL	Name F4 for list
Record format . . . . .	EMPLOY	Name F4 for list
File . . . . .	EMPLOYEES	Name *SRC F4 for list
Library . . . . .	EDISYSTEM	Name *CURLIB *LIBL
F3=Exit F4=Prompt F5=Refresh F12=Cancel F16=Remove keyword		

5、将光标放置在刚刚生成的字段 TOTALX 使用 F23 编辑字段显示格式：

Columns . . . :	1	71	Design Report	XXXXXXXX/QDDSSRC
RLU==>				LPRTF01
BASE	...+... 1	...+... 2	...+... 3	...+... 4
	...+... 5	...+... 6	...+... 7	
----- Start of Page 001 -----				
FLD1	<.....>;			<...>;<...>;
HEADER	职工信息			Page: 9999
FLD1	<.....>;	<.....>;	<.....>;	
<.....>;				
00002 +	雇员码	部门码	雇员名	小计
FLD1	<.....>;	<.....>;	<.....>;	<.....>;
00003 +	-----	-----	-----	-----
FLD1	<...>;	<.....>;	<.....>;	<.....>;
DETAIL	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00005 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00006 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00007 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00008 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00009 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
FLD1	<.....>;			<.....>;
TOTALN				999999999999999
----- End of Report -----				
F3=Exit F11=Define field F16=Delete field				
F22=Alternative keys F24=More keys				

6、键入 2 定义编辑码 EDTCDE，键入执行，将编辑码定义为 0，然后键入 2 次执行：

Specify Edit Code	
Keyword . . . . .	EDTCDE
Field . . . . .	TOTALX
Record format . . . . .	TOTALN
Type choices, press Enter.	
Edit code . . . . .	0
	1-9, A-D, J-Q, W-Z
Fill character . . . . .	*
	Currency symbol
F3=Exit F5=Refresh F12=Cancel F16=Remove keyword	

7、使用 Insert 插入/Delete 删除键向左/右调整 TOTALX 字段位置。最后完整的报表设计完成

Columns . . . :	1	71	Design Report	XXXXXXXX/OQDSSRC
RLU==>;				LPRTF01
BASE	...+... 1	...+... 2	...+... 3	...+... 4
	...+... 5	...+... 6	...+... 7	
	----- Start of Page 001 -----			
FLD1	<.....>;			<...>;<...>;
HEADER	职工信息			Page: 9999
FLD1	<.....>;	<.....>;	<.....>;	<.....>;
00002 +	雇员码	部门码	雇员名	小计
FLD1	<.....>;	<.....>;	<.....>;	<.....>;
00003 +	-----	-----	-----	-----
FLD1	<...>;	<.....>;	<.....>;	<.....>;
DETAIL	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00005 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00006 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00007 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00008 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00009 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
FLD1	<.....>;			<.....>;
" TOTALN				9 999,999,999,999,999.99 "
	----- End of Report -----			
F3=Exit	F11=Define field	F16=Delete field		
F22=Alternative keys	F24=More keys			

#### 6.2.2.7 保存报表设计名生成打印文件 PRTF

1、按 F3 退出，选择生成打印文件 Create printer file，改变默认值 Change defaults，可以选择生成原型报告 Prototype report。

Exit RLU		
Type choices, press Enter.		
Option . . . . .	1	1=Save and exit 2=Exit without saving 3=Resume
Member . . . . .	LPRTF01	Name
File . . . . .	OQDSSRC	Name
Library . . . . .	EDISYSTEM	Name
Text . . . . .	Demo for Design Report	
Create printer file . . . . .	y	Y=Yes, N=No
Change defaults . . . . .	y	Y=Yes, N=No
Prototype report . . . . .	y	Y=Yes, N=No
Change defaults . . . . .	N	Y=Yes, N=No
Submit to batch . . . . .	N	Y=Yes, N=No
Job description . . . . .	*USRPRF	Name, *USRPRF, *RLU
Library . . . . .		Name, *LIBL, *CURLIB
F5=Refresh	F12=Cancel	

2、键入执行，转到生成打印文件界面 Create Printer File (CRTPRTF)，再使用 F10 修改附加值，横向压缩打印字符：在生成打印文件选项将每英寸 10 个字符改为 15 个字符，按 A3 尺寸的纸型设计时，纸宽为 12 英寸，每行打印 198 或者 204 个字符(与具体打印机有关)，15CPI（每英寸字符数） $\times 198 \approx 12$  英寸。

Create Printer File (CRTPRTF)				
Type choices, press Enter.				
Additional Parameters				
Page size:				
Length--lines per page . . . . .	66		.001-255.000	
Width--positions per line . . . >	198		.001-378.000	
Measurement method . . . . .	*ROWCOL		*ROWCOL, *UOM	
Lines per inch . . . . .	6		6, 3, 4, 7.5, 7.5, 8, 9, 12	
Characters per inch . . . . .	15		10, 5, 12, 13.3, 13.3, 15...	
Front margin:				
Offset down . . . . .	*DEVD		0-57.790, *DEVD	
Offset across . . . . .			0-57.790	
Back margin:				
Offset down . . . . .	*FRONTMGN		0-57.790, *FRONTMGN, *DEVD	
Offset across . . . . .			0-57.790	
Overflow line number . . . . .	60		1-255	
				More...
F3=Exit	F4=Prompt	F5=Refresh	F12=Cancel	F13=How to use this display
F24=More keys				



3、系统提示已经成功生成了打印文件 LPRTF01。

我们可以使用 PDM 选项 19=Change using RLU 对其源文件进行修改。

当然，我们也可以通过 2 直接对源码进行修改；除非有特别需要，对于 RLU，我们不提倡直接对源码进行修改。

Work with Members Using PDM				XXXXXXXX
File . . . . .	QDDSSRC			
Library . . . . .	XXXXXXXX	Position to . . . . .		
Type options, press Enter.				
16=Run procedure	17=Change using SDA	19=Change using RLU		
25=Find string	54=Compare	55=Merge ...		
Opt	Member	Type	Text	
19	LPRTF01	PRTF	Demo for Design Report	
	MASCDP	PF	SYSTEM CODE DETAIL	
	MCITAP	PF	<V1.3 >; ITEM MASTER	
	MCITCL5	LF	<XLLC>;	
	MCITCP	PF	PROCESS SEQUENCE ITEM MASTER	
	MENUPGM	PF	EDI System Pulldown Menu PF	
	MJR84TJ	PRTF	ITEM RECEIPT PAGESIZE(51 132 *ROWCOL) OVRFLW(51)N	
	REFILE	PF	Data Dictionary Reference File (exercise)	
				More...
Parameters or command				
===>;				
F3=Exit	F4=Prompt	F5=Refresh	F6=Create	
F9=Retrieve	F10=Command entry	F23=More options	F24=More keys	



4、我们可以通过 SPLF 检查编译出现的问题和表样输出，对于编译错误，我们可以通过 SPLF 编译列表检查错误并修正。最终生成的报表样本如下：

列 . . . :	1	71	设计报告	XXXXXXXX/ODDSSRC
RLU==>;				LPRTF01
BASE	...+... 1	...+... 2	...+... 3	...+... 4
	...+... 5	...+... 6	...+... 7	
	----- 开始页码 001 -----			
FLD1	<.....>;		<...>;<...>;	
HEADER	职工信息		Page: 9999	
FLD1	<.....>;	<.....>;	<.....>;	<....>;
00002 +	雇员码	部门码	雇员名	小计
FLD1	<.....>;	<.....>;	<.....>;	<.....>;
00003 +	-----	-----	-----	-----
FLD1	<...>;	<.....>;	<.....>;	<.....>;
DETAIL	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00005 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00006 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00007 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00008 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
00009 S	XXXXX	XXXXXXX	XXXXXXXXXXXXX	999999999999999
FLD1				<.....>;
TOTALN				999,999,999,999,999.99
	----- 报告结束 -----			
F3= 退出	F11= 定义字段	F16= 删除字段		
F22= 备用键	F24= 其余键			

### 6.2.3 报表 PRTF 文件的源码

```

A*****
A          R HEADER
A*****
A          FLD005          10A  0    23CHRSIZ(2.0 2.0)
A                               SKIPB(1)
A          FLD010          5A  0    +26
A                               +0PAGNBR
A          FLD008          9A  0    4SPACEB(1)
A                               +3' 部门码'
A                               +8' 雇员名'
A                               +12' 小计'
A          FLD011          10A  0    3SPACEB(1)
A          FLD012          11A  0    +3
A          FLD013          13A  0    +5
A          FLD014          15A  0    +6
A*****
A          R DETAIL
A*****
A                               SPACEB(1)
A          DPCODE      R          0    4REFFLD(EMPLOY/DPCODE +
A                               EDI SYSTEM/EMPLOYEES)
A          EYCODE      R          0    +6REFFLD(EMPLOY/EYCODE +
A                               EDI SYSTEM/EMPLOYEES)
A          EYNAME      R          0    +10REFFLD(EMPLOY/EYNAME +
A                               EDI SYSTEM/EMPLOYEES)
A          SUBTOTAL    R          0    +7REFFLD(EMPLOY/SUBTOTAL +
A                               EDI SYSTEM/EMPLOYEES)
A          R TOTALN
A                               SPACEB(1)
A          TOTALX      R    17S 20    48REFFLD(EMPLOY/SUBTOTAL +
A                               EDI SYSTEM/EMPLOYEES)
A                               EDTCDE(0)
A

```

## 第七章 RPG语言

### 7.1 基本概念

RPG 的全称: Report Program Generator

IBM 的 AS/400 小型机上的高级语言. REPORT PROGRAM GENERATOR, 程序运行效率很高.

RPG 起始是一个非常简单的面向事务处理的编程语言, 用被储存在系统 IBM709 和 360 型号的 20 针打孔卡中的数据而生成报表.

系统 3 由于它的硬式磁盘而带来了 RPG 2, 使它变成中型的 IBM 机器的标准应用程序语言。

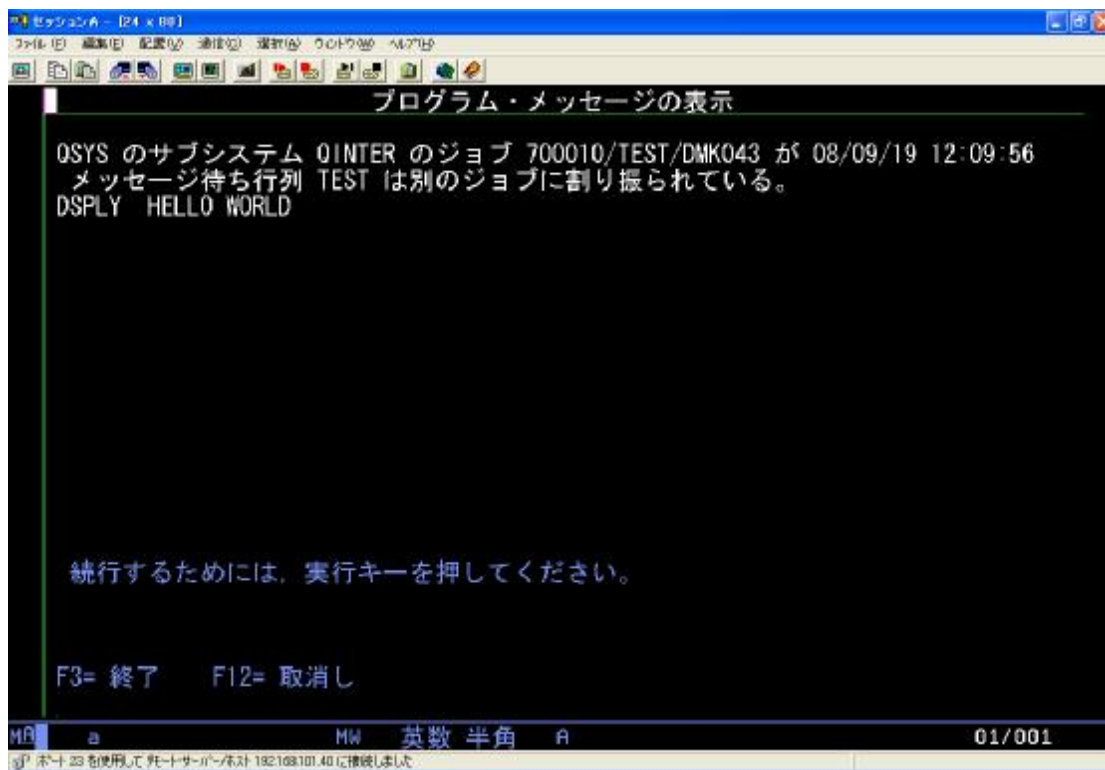
### 7.2 最简单的RPGLE 程序

为便于理解, 这里写一个最简单的RPGLE 程序

```
Columns . . . :   6 76           Edit           TST010/QRPGSRC
SEU==>                               LXPTTEST001
FMT *    *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+.
          ***** Beginning of data *****
0001.00 F*****
0002.00 C      'HELLO WORLD' DSPLY
0003.00 C              RETURN
          ***** End of data *****

F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
F16=Repeat find   F17=Repeat change   F24=More keys
(C) COPYRIGHT IBM CORP. 1981, 2002.
```

这个程序编译成功，并调用（CALL 程序名），就是在屏幕上反白显示“HELLO WORLD”字样。



与自由风格的C 语言不同，RPGLE 中的编码，是有一定的格式，如果写错，将会在当前代码行上高亮反绿显示。初学者如果不太清楚从何处开始下手，可以使用“F4”键查看（F4 键只有用2 进入的编辑状态才有效，用5 进入的查看状态是无效的）关于每一项所对应的内容代表什么意思，该如何填写，即如何写程序，将会在下面的具体讲解。

### 7.2.1 举例准备

列出表名，字段，以方便下面的举例。

假设有PF 文件叫PF01，文件的记录格式叫PF01R

每条记录，都是由DAT01、DAT02、DAT03 三个字段组成，每个字段都是两位长的字符型变量。

逻辑文件PF01L1 的键值为DAT 01

逻辑文件PF01L2 的键值为DAT 02

逻辑文件PF01L3 的键值为DAT 01、DAT 02

注：

文件的记录格式，可以理解为给这个文件整条记录起的一个名字；或者说将每条记录都视做一个类型相同大变量，然后给这个大变量起的名字。所以文件的记录格式信息中，包含有一条记录由多少个字段组成，总计长度是多少这样的信息。

文件的记录格式，与各个字段同时定义。（写文件的源码时）

文件的记录格式在RPGLE 的程序中，不能与文件名相同。



物理文件PF:

Columns . . . :	1	71	Edit	TST010/QDDSSRC
SEU==>				PF01
FMT PF	.....A.....	T. Name+++++	RLen++TDpB.....	Functions+++++
	*****	Beginning of data	*****	*****
0001.00	A	R PF01R		
0002.00	A*			
0003.00	A	DAT01	2A	
0004.00	A	DAT02	2A	
0005.00	A	DAT03	2A	
	*****	End of data	*****	*****
F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle				
F16=Repeat find F17=Repeat change F24=More keys				
(C) COPYRIGHT IBM CORP. 1981, 2002.				

逻辑文件LF:

Columns . . . :	1	71	Edit	TST010/QDDSSRC
SEU==>				PF01L1
FMT LF	.....A.....	T. Name+++++	Len++TDpB.....	Functions+++++
	*****	Beginning of data	*****	*****
0001.00	A	R PF01R		PF1LE(PF01)
0002.00	A*			
0003.00	A	K DAT01		
	*****	End of data	*****	*****
F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle				
F16=Repeat find F17=Repeat change F24=More keys				

## 7.2.2 简单的程序流程

```
Columns . . . : 6 76      Edit      TST010/QRPGSRC
SEU==>                                LXPTEST002
FMT FX FFilename++IPEASF. . . . L. . . . A. Device+. Keywords+++++
***** Beginning of data *****
0002.00 FPF01      UF   E      DISK
0002.01 C*
0003.00 C          READ      PF01R      90
0004.00 C*
0005.00 C          EVAL      DAT01=' 01'
0007.00 C*
0008.00 C N90      UPDATE      PF01R
0009.00 C*
0010.00 C          SETON                                LR
0011.00 C          RETURN
***** End of data *****

F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
F16=Repeat find      F17=Repeat change      F24=More keys
```

这个程序的意思，是说读PF01 这个文件，然后将读到的第一条记录中的DAT01 这个字段的值修改为“ 01” 。

“ SETON LR” ， LR 的位置可在HI、LO、EQ 中任选一处。意思是指将打开指示器\*INLR，即赋值使指示器\*INLR 的值等于1。等价于 “ EVAL \*INLR=' 1' ” ，意思是强制将内存中的数据写到磁盘中。（基于效率因素，系统在修改文件时，会先将修改的结果先放在内存中，在同一程序中，读取数据也是先从内存中查询。）LR，取自是Last Record

RETURN，表示程序结束，在后面“ 操作码” 一节中，会有讲述。

如果不太明白，就记住

```
C          SETON          LR
C          RETURN
```

或

```
C          EVAL          *INLR=' 1'
C          RETURN
```

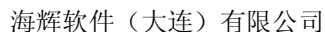
这两句话加在一起，表示程序结束就可以了。

从这个程序中，我们可以看到，RPGLE 的程序，大致上可以分为两个部分：

- 1、 声明、定义部分：声明程序中使用到的文件（F 行），定义程序中使用的变量（D 行）
- 2、 程序运行部分：即C 行，也就是程序段。

在RPGLE 程序中，F 行必须在D 行前面，D 行必须在C 行前面。

程序执行的起始顺序，将从定义部分之后，第一个C 行开始，顺序向下执行。



### 7.2.3 常见的程序流程

“\*”后面的，只是简单的解释，如果自己动手写，不需要输入这些内容。

1. 首句EVAL 赋值语句，直接执行；
2. 当系统发现操作码“ EXSR” 时，根据后面的变量名“ SUB#UPD” ， 去查找对应的“ SUB#UPD BEGSR” 语句；



3. 然后从“ SUB#UPD BEGSR” 之后，顺序向下执行，直至“ ENDSR” 语句

4. 执行到“ ENDSR” 之后，将会再回到当初的“ EXSR SUB#UPD” 处，继续向下执行，直到RETURN 语句为止这里提出一点要注意，如果子过程中，又执行了自身，即在SUB#UPD 程序中，又出现了“ EXSR SUB#UPD” ，是可以编译通过的，但在执行过程中，系统会因为无法定位，而出现死循环，直至报错异常中断退出。也就是RPGLE 的程序中，子过程不允许出现递归。

### 7.3 程序代码行的编写

#### 7.3.1 F 行说明

首位填上F，然后按F4，会出现如下内容：

Columns . . . :	6 76	Edit	TST010/QRPGSRC
SEU==>			LXPTEST003
FMT FX	FFilename++IPEASF. . . . L. . . . A. Device+. Keywords+++++		
	***** Beginning of data *****		
0010.00	FPF01 UF E DISK		
0011.00	DLSFLD01 S 2		
0012.00	C*		
0013.00	C EVAL LSFLD01=' 01'		
0014.00	C EXSR SUB#UPD		
Prompt type . . .	FX	Sequence number . . .	0010.00
Filename	File Type	File Designation	End of File
PF01	U	F	
File	Limits	Record	
Format	Processing	Address Type	Device
E			DISK
Keywords			Comment
F3=Exit	F4=Prompt	F5=Refresh	F11=Previous record
F12=Cancel		F23=Select prompt	F24=More keys

各项的含义分别是：

#### Filename:

需要声明的文件名，必须顶格，文件名必须唯一，也就是程序中对同样的文件名不能声明两次。

#### File Type:

声明文件的处理类型。必须填写。允许的选项有：

I： 输入型，即只读文件，对声明的文件只取其记录的值，不对记录进行修改

U： 修改型，即对声明的文件进行修改操作（删除记录属于修改操作的一种）

O： 输出型，即只写，对声明的文件只进行写操作。

C： 混合型，用于对屏幕文件的定义。（混合型，即输入/输出型，以屏幕文件为例，也就是读取屏幕文件的一些输入字段信息，同时也可以输出一些字段的值到屏幕文件中，但不能对屏幕文件

自身进行修改，所以与上面的U 是有区别的)

#### File Designation:

文件的指定方式，允许的选项有：

当“ File Type” 为I, U, C 时，这里填“ F”

当“ File Type ”为0 时，这里不填写

#### End of File:

程序结束前，对记录的处理方式。可以不填，或填“ E” 。似乎不填，表示在程序结束前，要处理所有文件的所有记录（含LF）；填E，表示只处理这个文件的所有记录。

总之，此项一般是不填。

#### File Addition:

是否会增加文件中的记录，即是否会对文件进行写操作。

可以不填，或填“ A”

当File Type 为“ 0” 时，系统自动默认此项为“ A” ，不必填写；

当File Type 为“ I” ，或“ U” 时，这项内容可以填“ A” ，也可以不填。不填，即表示不会增加文件中的记录，也就是没有写操作；填“ A” 时，即表示会增加文件中的记录，也就是会对文件进行写操作。

#### Sequence:

针对cycle 使用的，表示排序顺序。

当定义为非cycle 文件时，即“ File Designation” 项非“ P” 、 “ S” 时，此项必须为空；

当定义为cycle 文件时，即“ File Designation” 项为“ P” 、或“ S” 时，此项可填空、A、D。

A 表示升序，D 表示降序。

因为CYCLE 现在已不常用，所以通常不填。

#### File Format:

文件格式，不能为空，允许的值有：

E： 声明的文件，是外部描述的文件（即文件在程序运行之前就已存在）

F： 声明的文件，是一个程序描述文件

这里通常填“ E” ，即为外部描述文件

#### Record Length

“ File Format” 为“ F” 时，才需要填写。

通常不填

#### Limit Processing

通常不填。

#### Length of Key Field

查询时，索引键值的长度

如果“ File Format” 项等于“ E” ，即外部描述文件时，此项不填

如果“ File Format” 项等于“ F” ，便不需要按KEY 值查询时，此项也不填

如果“ File Format” 项等于“ F” ，需要按KEY 值查询时，此项填写KEY 值的长度

(1—2000)。

因为一般都使用外部描述文件，所以这里一般都不填写。

### Record Address Type

记录寻址类型，好象是对文件键值的描述。允许的值如下：

空：不使用KEY 值，在程序段中，不会对文件的查询定位操作，如“ SETLL”、“ CHAIN” 操作码都不会用的时，该项填空。

K： 使用KEY 值，即表示会对声明的文件进行查询定位操作，此时声明的文件必须有键值，即必须为逻辑文件（LF 文件），或在生成文件时，已加入了KEY 值。

（下面的选项应该是程序描述文件才会使用）

A： KEY 值为字符型

D： KEY 值为日期型

F： KEY 值为数字型

G： KEY 值为非英文字符

P： KEY 值为压缩型数字

T： KEY 值为时间型

Z： KEY 值为timestamp

总之，如果要按照键值对声明的文件进行查询定位操作（即程序中使用了CHAIN、SETLL 操作码，则此项需要填写“ K” ；如不需要进行查询操作，则不填。），此项填“ K” 时，声明的文件必须含有KEY 值。

### File Organization

一般不填

### Device

声明文件的存放位置，必须填写，允许的值有：

DISK： 磁盘文件，即文件存储在磁盘上，最常见的；

PRINTER： 打印文件，提供打印输出描述，以及对打印设备访问。打印报表用这个；

WORKSTN： workstation，工作站，显示文件。屏幕文件(DSPF)的定义用这个值

### Keyword

可以不填，常用的值有（这里只列出几个常用的）：

COMMIT

该文件记录的数据操作进行日志处理

RENAME

对文件记录格式名进行重命名。比如说程序中需要同时声明PFFHSL1，PFFHSL2这两个逻辑文件。这两个逻辑文件的记录格式名都是一样（通常和PF 一样，即都为FMTFHS；不过也可以定义成不同。如果不同，当然就不需要使用RENAME 键字了）。那么，为了能让系统区分，就必须对其中一个的记录格式名进行重命名。RENAME的语法：RENAME（旧记录格式名：新记录格式名），如下：

FPFFHSL1 IF E DISK

FPFFHSL2 IF E DISK RENAME (FMTFHS: FMTFHS2

新记录格式可以自由定义，只要在该程序中无同名的即可。RENAME 并不会真正的更改文件的记录格式名，仅是在当前运行程序中进行重命名。对同时运行的其它程序无影响

USROPN

对于声明的文件，由用户自行打开。如果不填写此关键字，系统将会在程序最最开始（执行第一句C 行语句前），自动执行“ OPEN 文件” 的操作，在程序结束后，自动执行“ CLOSE 文件” 的操作。而填写此关键字之后，OPEN, CLOSE 的操作将由用户在C 行程序段中，自行处理。如果用户未执行OPEN 操作，就执行CHAIN、READ、SETLL 等语句，在编译程序时就会报错。程序在结束之前，必须关闭所有已打开的文件，所以用起来会比较繁琐。USROPN 常作用于对文件的解锁，在同一程序中打开同一文件的不同MEMBER 等，属于一个较高级的用法，可在实际操作中慢慢体会。OPEN, CLOSE 的操作码，对应的是文件名，不是记录格式名。即

C OPEN PFFHSL1

C CLOSE PFFHSL1

而不是

C OPEN FMTFHS

#### Comment

注释说明。源自RPG，在RPG 中是有作用的，可以对程序作简短的说明，但在RPGLE 中，其实已经没有作用了，此项不用填。（填了也没用）

#### 常用例子

对文件进行只读的声明：

FPFFHS IF E DISK

对文件进行修改的声明：

FPFFHS UF E DISK

对文件进行只写的声明：

FPFFHS O E DISK

对文件进行修改，以及增加记录的操作：

FPFFHS UF A E DISK

对文件进行查询，增加记录的操作，并对文件进行查询操作：

FPFFHSL1 IF A E K DISK

声明两个记录格式相同的文件，并对其中之一进行重命名

FPFFHSL1 IF E K DISK

FPFFHSL2 IF E K DISK RENAME(FMTFHS: FMTFHS2)

注：在声明时，两个文件不一定要上下紧接着；随便改哪一个文件对应的记录格式都可以；新旧记录格式名用冒号隔开，新记录格式名可自行定义，无规则。

对文件的修改操作进行日志处理：

FPFFHSL2 UF E K DISK COMMIT

cycle 类文件的声明:

```
FPFFHSL2 IP E K DISK
```

这样文件声明为P 之后，程序中不需要写循环读文件，也不需要写RETURN，设指示器INLR，也就是

```
FPFFHSL2 IP E K DISK
```

```
C READ 记录格式名
```

等价于

```
FPFFHSL2 IF E K DISK
```

```
C DOW 1 = 1
```

```
C READ 记录格式名 EQ 指示器
```

```
C IF EQ指示器=' 1'
```

```
C LEAVE
```

```
C ENDIF
```

```
C ENDDO
```

```
C RETURN
```

### 7.3.2 D 行说明

首行填“ D” ，然后按F4，会出现如下内容：

Columns . . . :	6	76	Edit	TST010/QRPGSRC	
SEU==>				LXPTST003	
FMT FX	FFilename++IPEASF. . . . L. . . . A. Device+. Keywords+++++				
	***** Beginning of data *****				
0010.00	FPF01	UF	E	DISK	
0011.00	DLSFLD01		S	2	
0012.00	C*				
0013.00	C		EVAL	LSFLD01=' 01'	
0014.00	C		EXSR	SUB#UPD	
0015.00	C*				
Prompt type . . .	D	Sequence number . . .		0011.00	
Declaration					
Name	E	S/U	Type	From To / Length	
LSFLD01	-		S		2
Internal		Decimal			
Data Type		Positions	Keywords		
Comment					
F3=Exit F4=Prompt F5=Refresh F11=Previous record					
F12=Cancel F23=Select prompt F24=More keys					

## 内容说明

## Name:

定义的变量的名字，该名字可以不顶格写。（即允许有缩进）

## E

如果是程序内部自行定义一个临时变量，此处不填；

如果是引用的一个外部文件作为数据结构，那么这里就要填“ E”；同时“ Declaration Type”处，就要填“ DS”，即定义为一个结构；“ Keywords”处要使用EXTNAME 关键字

所谓“引用一个外部文件作为数据结构”，也就是说定义一个结构，整个结构中的变量，参照外部文件来定义。

所谓结构，可以理解为一个“由多个变量组合而成的大变量”。

举例而言：

```
D MYDS E DS EXTNAME(PFFHS)
```

和

```
D MYDS DS
```

```
D FHS01 1 2 (1 在From 项; 2 在To / length 项)
```

```
D FHS02 3 4
```

```
D FHS03 5 6
```

是等价的，都是定义一个结构变量MYDS（名字可以自行定义），这个结构变量是由三个字符型变量FHS01，FHS02，FHS03 拼成的。

第一种定义方法，就是引用外部文件“ PFFHS”作为数据结构的定义，注意使用到了“ EXTNAME”关键字，而且“ E”项的值为“ E”。

而第二种定义方法，就是直接定义一个结构“ MYDS”。注意没有使用外部文件时，“ E”项的值为空。

## S/U

一般都填空。

## Declaration Type

定义变量的类型，允许的值如下：

不填：非以下内容：数据结构、常量、独立变量、数组、表。此项为空时，好象只能用来表示当前定义的变量是属于结构的一个变量。在下面会举例

DS：数据结构，即定义一个结构变量，这个之前已讲过

C：常量

常量只能使用字符，不需要定义常量的长度、类型。常量的内容写在“ Keywords”处，并使用CONST 关键字，在程序段中，不能对常量进行赋值操作。

```
D MYNUM C CONST('abcdefghijklmn')
```

就是定义一个叫做MYNUM 的常量，这个常量包含字母a--n。

S：定义以下内容：独立变量、数组、表

定义一个叫MYFIELD1 的变量，变量为1 位长的字符型

D MYFIELD1 S 1 //1在“ To/length” 项

定义一个叫MYARRAY 的数组，共含3 条记录，每条记录为1 位字符型

D MYARRAY S 1 DIM(3) //DIM在“ Keywords” 项表的定义没有用过

总之，这一项，最常用的，就是“ DS”、“ S” 与空。即结构体与独立变量，其它选项较少用到。

### From

当“ Declaration Type” 项为“ S” 时，表示独立变量、数组，此项不填

当“ Declaration Type” 项为“ DS” 时，表示结构，此项仍然不填

当“ Declaration Type” 项为空时，表示当前定义的变量，属于上面定义的结构，此时可以填写，也可以不填写。

当填写时，“ From” 项表示变量在结构中的起始位置，右对齐；“ To/length” 表示变量在结构中的结束位置，也是右对齐。

当不填写时，“ To/length” 表示直接定义为变量长度。

举例：

D MYDS DS

D DSFLD01 1 2 //1 在“ From” 项，2 在“ To/length” 项

D DSFLD02 3 4

与

D MYDS DS

D DSFLD01 2 //2 在“ To/length” 项

D DSFLD02 2

其实是等价的，都是定义一个结构变量MYDS，这个结构变量中，包含了两个变量DSFLD01，DSFLD02，这两个变量都是两位长字符。所不同的是，第一种定义方法，是指定了变量在结构中的位置；而第二种方法，是直接指定变量的长度和类型

注意到上面的定义中，DSFLD01、DSFLD02 的Declaration Type 为空，也就是表示这两个字段是属于上面定义的结构MYDS。如果此项为“ S” ，即表示这个变量与结构无关

D MYDS DS

D DSFLD01 2 //2 在“ To/length” 项

D DSFLD02 S 2

在这个定义中，变量DSFLD02 就是一个独立的变量，与结构MYDS 无关。

### Length

上面已讲述在定义结构时的使用方法。

在定义非结构时，此项的内容即为定义变量的长度。右对齐

### Internal Data Type

定义变量的类型，允许的值有：

空：变量定义为字符型、压缩型数字

A： 变量定义为字符型

B： 二进制？不知道

- D: 变量定义为日期型
- F: 变量定义为浮点型
- G: 变量定义为图型（非英文, 汉字）
- I: 变量定义为带符号的整数
- N: 变量定义为指示器变量（没用过）
- P: 变量定义为压缩型数字
- S: 变量定义为普通的数型
- T: 变量定义为时间型
- U: 变量定义为无符号的整数
- Z: 变量定义为日期+时间型（格式：年-月-日-时.分.秒.微秒）
- \*: 变量定义为指针型

其实我最常用，就是不填，因为一般的程序，有字符和数字这两种类型变量，就足够了。

### Decimal Positions

当变量定义为数字型时，用来标志小数的位数。

当“ To/Length” 项为3，“ Internal Data Type” 项为空时

此项为空，表示定义的变量为3 位长的字符型

D MYFLD01 S 3 //定义为3 位字符型

此项不为空（右对齐），表示定义的变量为数字型

D MYFLD01 S 3 2 //定义数字型变量，1 位整数，2 位小数（总长为3 位）

### Keywords

关键字，可以不填，常用的值如下：（同样，这里我也只列出几个常用的，这里先不做详细说明，仅供参考，在后面的例子，看看就知道用法了）

CONST: 定义常量的值

DIM: 定义数组

EXTNAME: 引用外部文件作为数据结构变量

EXTFLD: 对引用了外部文件作为数据结构的某个变量，进行重命名

LIKE: 定义变量时，参照已存在的变量定义

OCCURS: 定义结构体变量时，指定的结构体变量的记录条数

INZ: 定义变量时，赋值初始值

DATFMT: 定义日期变量时，指定日期格式

\*MDY (mm/dd/yy)

\*DMY (dd/mm/yy)

\*YMD (yy/mm/dd)

\*JUL (yy/ddd)

\*ISO (yyyy-mm-dd)

\*USA (mm/dd/yyyy)

\*EUR (dd.mm.yyyy)



---

\*JIS (yyyy-mm-dd)

常用例子

定义一个10 位长的字符型变量:

```
D MYFLD S 10
```

定义一个10 位长, 其中含2 位小数的字符型变量, 并使其初始值为1

```
D MYFLD S 10 2 INZ(1)
```

定义一个每条记录为5 位长字符型变量, 共10 条记录的数组

```
D MYFLD S 5 DIM(10)
```

定义一个10 位长的字符型变量, 再定义一个变量, 参照前一变量定义

```
D MYFLD01 S 10
```

```
D MYFLD02 S LIKE(MYFLD01)
```

定义一个结构, 由一个3 位长的字符变量, 和一个10 位长, 其中2 位小数的数字变量组成

```
D MYDS DS
```

```
D MYDS01 3
```

```
D MYDS02 10 2
```

定义一个结构变量, 结构内容参照外部文件PFFHS

```
D MYDS E DS EXTNAME(PFFHS)
```

定义一个结构变量, 结构内容参照外部文件PFFHS, 并且将第二个字段重命名为FHS999

```
D MYDS E DS EXTNAME(PFFHS)
```

```
D FHS999 E DS EXTFLD(FHS02)
```

定义一个日期型变量, 格式为yyyy-mm-dd

```
D MYDATE S D DATFMT(*ISO)
```

### 补充说明

变量的定义, 除了在D 行定义之外, 还可以在C 行通过赋值语句直接定义

如

```
D FLD01 S 2 INZ(' 01' )
```

与

```
C MOVE ' 01' FLD01 2 //2 在length 处, 右对齐
```

是等价的

定义结构之后, 可以将结构变量视为一个普通的变量进行赋值来改变结构变量的值, 也可以通过组成结构变量的变量进行赋值, 来达到修改结构变量的值的目的。

如:

```
D MYDS DS
```

```
D MYFLD01 2
```

```
D MYFLD02 2
```

在C 行中，这两句是等价的

```
C EVAL %SUBST(MYDS:3:2)=' 01'
```

```
C EVAL MYFLD02=' 01'
```

第一句是直接改结构变量MYDS 的后两位的值（当然，此时MYFLD02 的值也变化了）第二句是对MYFLD02 进行赋值，同样，赋完值之后，MYDS 的后两位也变为' 01' 在需要频繁进行数字与字符之间转换时，偷懒的人会通过定义这样的结构来达到目的：

```
D MYDS DS
```

```
D MYFLD01 1 8
```

```
D MYFLD02 1 8 0
```

比如说，给MYFLD01 赋值为' 20070208' 之后，MYFLD02 也就自动等于20070208；然后给MYFLD02 加1 之后，MYFLD02 等于20070209，MYFLD01的值也自动等于' 20070209' 。可以认为结构变量MYDS 是字符型（即一直等于MYFLD01 的值）

这种方法，当需要字符型变量时，就使用MYFLD01；当需要数字变量时，就使用MYFLD02，不过我总觉得有点类似于作弊，一般没用。

### 7.3.3 入口参数

程序可以通过“ \*ENTRY” 定义入口参数，或称之为接口参数，来传递数据。

假设有程序FHS01ILE，其中入口参数的定义如下：

```
C          *ENTRY PLIST
```

```
C          PARM          FLD01  3
```

```
C          PARM          FLD02  4
```

其中：

\*ENTRY 在“ Factor 1” 项；

PLIST 在“ Operation” 项；

PARM 在“ Operation” 项；

FLD01、FLD02 都在“ Result” 项上述定义，表示这个程序通过两个字段与其它外部程序沟通。

那么别的程序(如FHS02ILE)在调用程序FHS01ILE 时，就要带上两个字符型变量，如

```
C          CALL ' FHS01ILE'
```

```
C          PARM          FHSFLD01  3
```

```
C          PARM          FHSFLD02  4
```

在两个程序里，这两个变量名可以不同（比如说一边叫FHSFLD01，FHSFLD02；一边叫FLD01，FLD02），但长度，类型必须匹配。

如果在FHS02ILE 中，FHSFLD01 等于' 123' ，FHSFLD02 等于' abcd' ，那么系统在运行CALL 语句，执行程序FHS01ILE 时，将会对字段FLD01 初始化赋值，使其一开始就等于' 123' ，字段FLD02 等于' abcd' 。

如果FHS01ILE 程序中，对FLD01、FLD02 进行了改动，比如FLD01 最后等于' 789' ，FLD02 最后等于' efgh' ，那么程序FHS02ILE 在调用完FHS01ILE 之后，FHSFLD01、FHSFLD02 这两个字段也

同样会改变，成为' 789'，和' efgh'

也就是入口参数的变化是可以传递的。

入口参数的定义，可以写在程序的任何一处，而程序的执行，始终是从C 行的顺序第一行开始执行，与入口参数所在的位置无关。

FHS02ILE 也可以使用一个大变量来调用FHS01ILE，只要总长相等即可（这种方法仅限于被调用的程序FHS01ILE 的入口参数全部为字符型才可使用，仅仅只是不会错，不建议这样使用。

```
C      CALL ' FHS01ILE'
          PARM          FHSFLD01      7
```

其实从上面的例子可以看出，入口参数可以使用结构的形式来表达，所以下面这种写法也不会有错。（如果被调用程序有数字型变量，只要在定义结构时也定义为数字型即可）

```
D MYDS DS
D DS01 3
D DS02 4
C CALI ' FHS01ILE'
C PARM MYDS
```

不过要注意，如果RPG 程序调用C 程序，那么入口参数必须严格按照C 程序中的来，比如C 程序中带了两个字符型参数，那么RPG 程序中也必须是两个字段入口参数，不能使用由两个字符变量组成的结构。原理可以自行想想。

既然可以使用结构做为入口参数，当然，也可以参照外部文件来定义结构做为入口参数

```
D MYDS E DS EXTNAME(PFFHS)
C CALL ' FHS01ILE'
C PARM MYDS
```

与

```
C      CALL ' FHS01ILE'
C          PARM          FHS01      2
C          PARM          FHS02      2
C          PARM          FHS03      2
```

是等价的。

可以看到，参照外部文件定义结构做为入口参数时，可以有效的节省代码行，而且不会出现遗漏。所以在实际使用中，常会看到，将一些公共程序的入口参数定义成一个PF 文件。而调用它的程序，就参照这个PF 文件，定义结构做为调用的接口参数。

当接口参数不一致时，如FHS02ILE 中漏了第二个参数时：

```
C      CALL ' FHS01ILE'
C          PARM          FHS01 3
```

此时，并不是一开始运行FHS01ILE 程序，系统判断入口参数不符就报错；实际上，此时，FLD01 的值还是正确的，但FLD02 的值就处于一个未初始化的状态。于是，当代码执行到与FLD02 有关的操作码时，才会报错；如果FHS01ILE 在运行的过程中，因为逻辑判断（如IF 条件判断）的关系，

而未执行任何与FLD02 有关的操作码，那么程序会正常运行完毕，不会有报错。

这时，FHS02ILE 调用了程序FHS01ILE 之后，程序中原有的接口参数的数据就可能因为这次调用程序而发生错位，从而导致数据的错误、混乱。数据的错误、混乱其实还不是最大的问题，更大的问题在于“这时我们不知道数据已经出错了”。解决之道，也是如上所说，对于调用频繁，且入口参数较多的公共程序，考虑将其入口参数写成一个PF 文件。这样调整入口参数时，只要修改PF 文件并重新编译，再编译相关程序即可（至少发生遗漏时，程序会报错异常中断，不会出现错误的数

#### 7.3.4 C 行说明

Columns . . . :	6 76	Edit	TST010/QRPGSRC
SEU==>			LXPTTEST003
FMT FX	FFile name++IPEASF....L....A. Device+. Keywords+++++		
	***** Beginning of data *****		
0010.00	FPF01	UF E	DISK
0011.00	DLSFLD01	S	2
0012.00	C*		
0013.00	C	EVAL	LSFLD01=' 01'
0014.00	C	EXSR	SUB#UPD
0015.00	C*		
0016.00	C	EVAL	LSFLD02=' 02'
0017.00	C	EXSR	SUB#UPD
0018.00	C*		
Prompt type . . .	CX	Sequence number . . .	0013.00
Level	N01	Factor 1	Operation
			EVAL
Extended			
Factor 2			Comment
LSFLD01=' 01'			
F3=Exit	F4=Prompt	F5=Refresh	F11=Previous record
F12=Cancel		F23=Select prompt	F24=More keys

内容说明

#### Level

一般不填

#### N01

这个含义比较丰富，我只用过其中一种：

首位不带N，后面填写01—99 的数字时，表示相应的指示器打开时，执行后面的操作，

如：

```
C 12 EVAL FHS01=' 01'
```

等价于

```
IF *IN12=' 1'
```

```
EVAL FHS01=' 01'
```

```
ENDIF
```

首位带N，后面填写01—99 的数字，表示相应的指示器关闭时，执行后面的操作

要注意，该项内容仅作用于该行操作码。如果指示器打开后，需要执行多条语句，那么每条语句前面，该项都要赋值。

即

```
C IF *IN12=' 1'
```

```
C EVAL FHS01=' 01'
```

```
C EVAL FHS02=' 02'
```

```
C ENDIF
```

如果用这种方式来表达，就要写作

```
C 12 EVAL FHS01=' 01'
```

```
C 12 EVAL FHS02=' 02'
```

所以说，根据指示器状态来执行的语句，在执行少量操作码时，可以使用这种方法；如果语句较多，修改起来不方便，还是直接用IF—ENDIF 的判断语句比较合适。

#### Factor 1

操作内容一，将在后面与操作码一起讲

#### Operation

操作码，后面有专门章节讲解操作码

#### Factor 2

操作内容二，同上

#### Result

操作结果，同上

#### Length

长度。

变量的定义，除了在D 行定义之外，还可以在C 行通过赋值语句直接定义

如

```
D FLD01 S 2 INZ(' 01' )
```

与

```
C MOVE ' 01' FLD01 2 //2 在length 处，右对齐
```

是等价的

一个变量，在整个程序中，只要定义一次就可以了，对定义的顺序没有强制要求。

#### Decimal Positions

与length 相呼应，当此项有值时，表示定义的是一个数字型变量，该项表示小数位长度。

如

```
C Z-ADD 2 FLD02 3 2
```

即是说，将FLD02 定义为一个3 位长，其中1 位整数，2 位小数的数字变量，并赋值为2.00

#### HI、LO、EQ

这是三个指示器位置项。可赋值的内容是从01—99，在以后的说明中，如果HI 项填写10，L0 项填写20，EQ 项填写30，那么我所说的HI 指示器，即是指\*IN10，L0 指示器即是\*IN20，EQ 指示器即是\*IN30，依此类推。（也就是说，HI 指示器，并不是\*INH1，事实上，也没有\*INH1 这个指示器）

#### ILE 操作码分类：

##### 1. 程序流程控制

DO、DOU、DOUxx、DOW、DOWxx、ITER、LEVAE

IF、ELSE、ELSEIF、IFxx、ORxx、ANDxx

SELECT、WHEN、WHENxx、OTHER、ENDxx、GOTO、TAG、EXSR、BEGSR、ENDSR

CABxx

##### 2. 初始化操作

CLEAR、RESET

##### 3. 文件操作

OPEN、CLOSE、

CHAIN、SETGT、SETLL、READ、READC、READE、READP、READPE、DELETE、UPDATE、WRITE、  
UNLOCK

ROLBK、COMMIT、EXFMT、ACQ、EXCEPT、FEOD、FORCE、NEXT、POST、REL

##### 4. 程序调用

CALL、CALLB、CALLP、PARM、PLIST、RETURN

##### 5. 赋值语句

MOVE、MOVEA、MOVEL、EVAL

##### 6. 字符操作

CAT、CHECK、CHECKR、SCAN、SUBST、XLATE

##### 7. 数字操作

ADD、DIV（除）、MULT（乘）、MVR（除法取余）、SQRT（开方）、SUB、XFOOT、

Z-ADD、Z-SUB

##### 8. 数组操作符

LOOKUP、MOVEA、SORTA、XFOOT

##### 9. 数据区操作(没用过)

IN、OUT、UNLOCK

##### 10. 日期操作

ADDUR、EXTRCT、SUBDUR、TEST

##### 11. 指示器操作

SETOFF、SETON

##### 12. 信息操作（前两个没用过）

DUMP、SHTDN、TIME、DSPLY

##### 13. 内存管理操作(完全没用过)

ALLOC、DEALLOC、REALLOC

14. 位操作（没用过）

BITOFF、BITON、TESTB

### 7.3.5 ILE 操作码

#### 7.3.5.1 A—C开头

##### ACQ {(E)} (Acquire)

取地址位。其实400 的程序中也有指针型变量，那么也就会有地址位，这个命令是取地址位的。

##### ADD {(H)} (Add) 加法操作

1. 基本语法：

Factory 1	Operation	Factory 2	Result
FHS01	ADD	FHS02	FHS03 // RPG 的语法

等价于

EVAL FHS03=FHS01+FHS02 //RPGLE 的语法

FHS01、FHS02、FHS03 必须都为数字型变量(P 型,或S 型),或者就是数字意思是将Factory 1 项的数据,加上Factory 2 项的数据,赋值到Result 项上

2. 语法二：

如果这样写的话：

Factory 1	Operation	Factory 2	Result
	ADD	FHS02	FHS03

就等价于：

EVAL FHS03=FHS03+FHS02

即Factory 1 项未填时,就表示将Result 项的数据,加上Factory 2 项的数据,然后赋值到Result 项上

3. 四舍五入：

(H) 表示四舍五入,如FHS02=12.50 (4, 2), FHS03=3 (2, 0), 那么

ADD(H)	FHS02	FHS03
--------	-------	-------

执行之后, FHS03=16 (因为进位了)

而

ADD	FHS02	FHS03
-----	-------	-------

执行之后, FHS03=15

不过实际使用中,我们都尽可能使相加的字段小数位数相等,所以ADD 操作码一般都没有使用到四舍五入的功能。

4. 可以在ADD 操作时,对Result 项的变量进行定义

Factory 1	Operation	Factory 2	Result
FHS01	ADD	FHS02	FHS03 10 2

EVAL 语句不能对在赋值的同时,对变量进行定义。

## 5. 关于结果数据超长时的问题:

当加出的结果超长, 比如FHS03 定义为3, 2 (1 位整数, 2 位小数, 下同) 时, 再假设FHS01=10, FHS02=4。那么FHS01+FHS02 本来应该等于14, 但用ADD 操作之后, 系统会自动将超长位截去, 即FHS03 最后等于4; 而用EVAL 语句时, 系统判断超长后, 会直接报错 “ The target for a numeric operation is too small to hold the result”, 然后异常中断。

使用ADD 操作码程序不会异常中断, 但有可能发生了错误的数字我们也不知道; 使用EVAL 操作码可以避免产生错误的数字, 但程序会异常中断, 需要进行人工干预。可根据实际情况选择使用ADD 还是EVAL。

### ADDUR {(E)} (Add Duration) 日期时间相加

#### 1. 对日期型变量进行加操作, 比如说已定义日期型变量MYDATE1, MYDATE2, 将

MYDATE1 的日期加上3 天, 赋值到MYDATE2 中:

Factory 1	Operation	Factory 2	Result
MYDATE1	ADDUR	3: *D	MYDATE2

其中, Factory 1, Result 项, 都必须为日期型变量 (即在D 行, Internal Data Type 项为 “ D” )

#### 2. 与ADD 操作码相同, Factory 1 项为空时, 表示直接在Result 项上进行日期相加, 如将MYDATE1 直接加上3 个月 (即结果也是赋值到MYDATE1 中):

Factory 1	Operation	Factory 2	Result
	ADDUR	3: *M	MYDATE1

#### 3. 日期型变量的参数含义:

\*D表示天, 也可用\*HOURS

\*M表示月, 也可用\*MONTHS

\*Y表示年, 也可用\*YEARS

#### 4. 除了日期型之外, 还有时间型, 日期时间型, 都可以使用ADDUR 操作码.

在D 行, Internal Data Type 定义为 “ T”, 表示时间型 (时、分、秒)

Internal Data Type 定义为 “ Z”, 表示日期时间型 (年、月、日、时、分、秒、微秒)

在使用ADDUR 操作码, 时间型的参数如下:

\*H 表示小时, 也可用\*HOURS

\*MN 表示分钟, 也可用\*MINUTES

\*S 表示秒, 也可用\*SECONDS

而日期时间型, 除了可以使用\*Y、\*M、\*D、\*H、\*MN、\*S (以及相应的全称) 之外, 还可以对微秒进行处理, 参数为\*MS, 或\*MSECONDS

#### 5. Factory 2 项中的数字, 可以使用负数, 使用负数时, 表示减去相应的年、月、日, 不过通常会使用SUBUR 这个操作码来进行日期的减法, 语法与ADDUR 相同

### ANDxx (And) 条件判断语句— “ 与”



1. 在RPG 的用法中，有ANDEQ, ANDNE 之类的，与IF 语句一起联用。

Factory 1	Operation	Factory 2	Result
FLD01	IFEQ	' A'	
FLD02	ANDEQ	' B'	
。。。处理内容			
ENDIF			

2. AND 后面跟的xx，可以有如下用法：

EQ	Factory 1 = Factory2
NE	Factory 1 <> Factory2
GT	Factory 1 > Factory2
LT	Factory 1 < Factory2
GE	Factory 1 >= Factory 2
LE	Factory 1 <= Factory 2

#### BEGSR (Beginning of Subroutine) 子过程的开始处

Factory 1	Operation	Factory 2	Result
BEGSR	子过程名		

在前面，讲述程序流程时，已经对子过程进行了解释。这里，BEGSR 本身用法没什么特别的，只是要注意有BEGSR 语句，就一定要有ENDSR 对应。否则程序编译会报错。所以这里建议，如果是自己从头到尾写一个程序，最好写了BEGSR 语句后，马上写一个ENDSR，然后再来写中间的内容，避免遗漏。

在程序中，可以写一个子过程，但是不调用它。（也允许就是子过程没有相应的EXSR语句）也允许写一个空的子过程。（也就是BEGSR 语句与ENDSR 语句之间没有别的执行语句了）

#### CALL {(E)} (Call a Program) 调用外部程序

Factory 1	Operation	Factory 2	Result
CALL	' 外部程序名'		

1. 如果是直接调用外部程序，那么程序名称需要用单引号括起来，且该程序必须存在。如

CALL ' FHSILE01'

就表示调用“ FHSILE01” 这个外部程序

2. 如果没有用单引号，如

CALL FHSILE01

就表示，FHSILE01 这时是个字符型变量（即并非调用“ FHSILE01 这个程序），调用的是变量内容所代表的程序。如：

FHSILE01=' FHS01' 时，表示调用“ FHS01” 这个外部程序；

FHSILE01=' FHS02' 时，表示调用“ FHS02” 这外外部程序

也就是说，CALL 操作码调用的程序名，可以是一个变量名。

3. 被调用的外部程序如果有接口参数，那么CALL 操作码之后，也需要有“ PARM” 操作码相对

应。

4. 这一点要注意：虽然400 的程序段代码中，是不区分大小写；但调用的程序名，要区分大小写。即‘ FHSILE01’，与fhsi le01，表示的是两个不同的程序。在使用时要注意，尤其是程序名使用字符变量来表达时，可能会因为大小写的问题而导致 CALL 不到想CALL 的程序，从而导致程序异常中断。

#### CASxx (Conditionally Invoke Subroutine) 带条件的调用子过程

1. 表示根据xx 项对Factory 1 与Factory 2 进行判断，当符合条件时，执行Result 处的子过程。需要配合“ END” 或“ ENDCS” 语句来表示条件判断的结束。不需要“ SELECT” 操作码。
2. 举例如下：

Factory 1	Operation	Factory 2	Result
FLD01	CASEQ	‘ 1’	SUB01
FLD01	CASEQ	‘ 2’	SUB02
	CAS		SUB03
	ENDCS		

表示当FLD01 等于‘ 1’ 时，执行子过程SUB01；

当FLD01 等于‘ 2’ 时，执行子过程SUB02；

当不满足以上两个条件时，执行子过程SUB03

最后的“ ENDCS” 必须要有，表示条件判断结束；但不需要SELECT。上面这段语句，与下面这一段是等价的：

Factory 1	Operation	Factory 2	Result
	SELECT		
	WHEN	FLD01=‘ 1’	
	EXSR	SUB01	
	WHEN	FLD01=‘ 2’	
	EXSR	SUB02	
	OTHER		
	EXSR	SUB03	
	ENDSL		

3. 可以看出来，CASxx 这种语句，是用于逻辑判断仅一个条件时的分支处理，这样的写法在代码的阅读上会很直观。而当逻辑判断大于一个条件时，这个语句就不适用了。

#### CAT {(P)} (Concatenate Two Character Strings) 字符连接

## 1. 基本语法:

Factory 1	Operation	Factory 2	Result
FLD01	CAT	FLD02: 0	FLD03

这句话的意思,是将FLD02 拼在FLD01 后面,中间没有空格,然后将拼出的结果赋值到FLD03 中。

FLD02: 0, 表示FLD02 与FLD01 之间的空格数为0, 依此类推

FLD02: 1, 就表示FLD01 后面加一个空格, 再拼上FLD02

FLD02: 3, 就表示FLD01 后面加三个空格, 再拼上FLD02

Factory 1 项, 与Factory 2 项可以是字符型变量, 也可以就是字符。当是字符时, 需要用单引号将字符括起来

2. 其实根据RPG 的语法, Factory1 项如果不填值, 就表示将Factory 2 项的内容直接拼在 Result 项上, 这里就不举例了。

3. 字段FLD01 如果内容后面有空格, 如“ ABC ”, 那么在CAT 操作时, 系统会自动将后面的空格截去, 只取‘ ABC’。举例:

FLD01=‘ ABC ’ (8 位字符型),

FLD02=‘ 1’ (1 位字符型),

FLD03=‘ ’ (8 位字符型)

那么, 执行了下述语句之后

FLD01 CAT FLD02:0 FLD03

FLD03 就等于‘ ABC1 ’

而如果执行:

FLD01 CAT FLD02:1 FLD03

FLD03 就等于‘ ABC 1 ’ (C 与1 之间有一个空格)

4. 表示空格个数时, 可以使用数字型的变量来表达, 如

EVAL N=1

FLD01 CAT FLD02:N FLD03

5. CAT 操作码, 其实也可以通过EVAL 来实现部分。比如将FLD01 与FLD02 拼起来, 中间无空格, 赋值到FLD03 中, 也可以写做:

EVAL FLD03=FLD01 + FLD02

6. EVAL 操作码的优势, 在于可以在一行语句中, 就把多个字符拼在一起, 如:

EVAL FLD05=FLD01+FLD02+FLD03+FLD04

7. EVAL 操作码的不足之处, 在于只能进行最简单的拼接, 无法自动将字符后面的空格去除, 如:

FLD01=‘ ABC ’ (8 位字符型),

FLD02=‘ 1’ (1 位字符型),

FLD03=‘ ’ (8 位字符型)

在执行语句

EVAL FLD03=FLD01+FLD02

后，

FLD03=' ABC '

因为FLD01 是8 位，FLD03 也是8 位，在操作时，不能去掉ABC 后面的5 位空格，此时后面再拼FLD02 已无意义，所以FLD03 仍是' ABC '

### CHAIN {(N | E)} (Random Retrieval from a File) 按键值对文件记录进行查询定位

#### 1. 基本语法：

举例，对逻辑文件PFFHSL1 进行定位操作。逻辑文件PFFHSL1，是以FHS01 为键值，文件记录格式名叫FMTFHS

Factory 1	Operation	Factory 2	Result	HI	L0	EQ
FHS01	CHAIN	FMTFHS		17	18	

这个例子中，FHS01 应该是一个与文件PFFHSL1 中键值（FLD01）类型、长度都相等的一个字符型变量，或者字符。

Factory 2 项中，要填文件的记录格式名，而不是文件名

HI 指示器表示是否查询到相应记录，查询不成功时，打开HI 指示器

L0 指示器表示查询时，文件是否被锁。文件被锁时，打开L0 指示器

也就是说：

\*IN17=' 0' ，表示查询到了对象记录。

\*IN17=' 1' ， \*IN18=' 0' ，表示无相应的对象记录

\*IN17=' 1' ， \*IN18=' 1' ，表示查询时，文件被锁（不确定有没有相应的对象记录）

2. 用修改方式声明的文件，当查询成功后，对象记录被锁定，其它程序无法以修改的方式定位到当前对象记录上。（但可以用只读的方式定位）
3. L0 指示器，仅用于用修改方式声明的文件。对于只读方式声明的文件其实无意义
4. 如果用修改方式声明文件，但在L0 处未填写指示器，且有程序锁住对象记录时，当前程序会根据PF 中定义的WAITRCD(Maximum record wait time)参数，等待相应的秒数(如WAITRCD 处为10，即表示锁表时，等待10 秒；如果填的是\*IMMED，就表示不等待，一判断对象记录被锁就结束操作)；如果在等待时间内，对方仍未解锁，当前程序就会异常中断退出；如果L0 处填写指示器，那么程序就不会中断退出，且L0 指示器打开。
5. 当FHS01 键值，在文件PFFHSL1 中，对应有多条记录时（即键值不唯一），程序将会按照文件的内部记录号由小到大的顺序，定位到符合条件的第一条记录。
6. 当一个逻辑文件，KEY 值有多项时，可以使用KLIST 操作码先定义一个组合键值，然后再用这个组合键值来进行CHAIN 操作。需要注意，组合键值中，必须每一个成员字段都与对象记录所对应的字段相等，才能查询成功。（所以组合键值，通常使用SETLL 定位较多）
7. 当用修改方式声明文件，但希望进行不锁记录的查询操作时，可以将CHAIN 操作码写为CHAIN (N)，这个括号(N)，就表示当前的查询定位，不对记录进行锁定操作（也就是用只读的方式来处理对象记录，所以只能取出对象记录的信息，不能做修改；如果要修改对象记录的话，

还必须进行一个CHAIN 操作)

#### 8. 这一条要特别注意:

当没有填写HI 指示器时,RPGLE 是允许编译通过的。而在某些情况之下,运行程序时,如果CHAIN 操作成功,找到了符合条件的记录时,没有任何问题;但如果CHAIN 操作没有找到符合条件的记录时,实际上系统会按照键值的排序以及内部记录号,定位到下一条记录上,这时再取出来的数据,就统统都是下一条记录的数据。所以,除非有绝对的把握,CHAIN 操作肯定成功,否则就一定要养成填写HI 指示器的良好习惯。

### CHECK {(E)} (Check Characters) 检查对象变量中的字符

#### 1. 基本语法:

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
FLD01	CHECK	FLD02: 2	N			42

语句CHECK 的意思,是指字段FLD02 中,从第二位字符开始,向右查找(含第二位字符),是否包含有FLD01 中没有的字符。

如果的确有在FLD01 中没有的字符,将字符所在位置赋值到变量N 中,同时打开EQ 指示器;如果从第二位字开始,向右读取的所有字符,都可以在变量FLD01 中找到,那么N 为0, EQ 指示器处于关闭状态。

“ FLD02: 2” 表示从变量FLD02 的第二位开始,向右作比较。如果仅仅只是

“ FLD02” , 那么就表示从变量FLD02 的首位开始,向右查找

#### 2. 实例

假设FLD01 为8 位长字符,且当前值为' 12345678'

而FLD02 为5 位长字符,且当前值为' A3456'

那么执行上述CHECK 操作后,N=0, \*IN42=' 0' (从第二位开始,3456 都存在于变量FLD01 中)

—假设FLD01 为8 位长字符,且当前值为' 12345678'

而FLD02 为5 位长字符,且当前值为' 34ABDC'

那么执行CHECK 操作后,N=3, \*IN42=' 1' (即第三位“ A” , 不存在于变量FLD01 中)

### CHECKR {(E)} (Check Reverse) 反向检查对象变量中的字符

#### 1. 基本语法:

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
FLD01	CHECKR	FLD02: 3	N			42

语句CHECKR 的意思,是指字段FLD02 中,从第三位字符开始,向左查找(含第三位字符),是否包含有FLD01 中没有的字符。(基本与CHECK 类似,所不同的,就是CHECK 是自左向右的查找;而CHECKR 是自右向左查找)

如果有,将字符所在位置赋值到变量N 中,同时打开EQ 指示器;

如果从第二位字开始,向左读取的所有字符,都可以在变量FLD01 中找到,那么N 为0, EQ

指示器处于关闭状态。

“ FLD02: 3” 表示从变量FLD02 的第三位开始，向左作比较。如果仅仅是

“ FLD02” ，那么就表示从变量FLD02 的末位开始，向左查找

## 2. 实例

假设FLD01 为8 位长字符，且当前值为‘ 12345678’

而FLD02 为5 位长字符，且当前值为‘ 23A56’

那么执行上述CHECK 操作后，N=0， \*IN42=‘ 0’ （从第三位开始，向左，23 都存在于变量FLD01 中）

假设FLD01 为8 位长字符，且当前值为‘ 12345678’

而FLD02 为5 位长字符，且当前值为‘ BC3B45’

那么执行CHECK 操作后，N=2， \*IN42=‘ 1’ （即第二位“ C” ，不存在于变量FLD01中）

## 3. 计算字符实际长段

根据CHECKR 的这个操作码的用法，我们可以使用它来取出变量截去尾部空格后的实际长度：

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
‘	CHECKR	FLD02	N			42

Factory 1 项，是一个空格字符。

N 表示的意思是：从变量FLD02 的尾部，向前数第一个非空格字符，在FLD02 中所处的位置。

那么这个N 当然就是变量FLD02 的实际长度；

如果指示器未打开，那么说明整行都是空，实际长度为0，也没错。

## CLEAR (Clear) 清除内容

### 1. 基本语法

Factory 1	Operation	Factory 2	Result
	CLEAR	对象名	

这个对象名，可以是程序中定义的结构、文件的记录格式名。

所谓文件的记录格式名，包括了程序中声明的磁盘文件、打印报表文件、屏幕文件CLEAR 操作的意思，就是将对象所对应的所有变量/字段都赋上空值。

## CLOSE {(E)} (Close Files) 关闭文件

### 1. 基本语法

Factory 1	Operation	Factory 2	Result
	CLOSE	对象文件名	

2. CLOSE 所对应的，是文件名，不是文件的记录格式名，要注意

3. CLOSE 操作码，仅适用于声明文件时，keyword 使用“ USROPN” 关键字的文件

4. 每一个CLOSE 的操作，在之前都必须有OPEN 文件的操作。也就是，文件必须打开了之后，才能关闭。不能关闭未打开的文件

5. 允许使用\*ALL 变量，来表达关闭所有已打开的文件：

CLOSE \*ALL

**COMMIT {(E)} (Commit) 日志事务处理的确认操作**

## 1. 基本语法

Factory 1 Operation Factory 2 Result

COMMIT

2. 该操作码无其它参数，就是指对事务处理进行确认操作。

3. ILE 程序中，COMMIT 操作可随时进行，也允许在没有声明COMMIT 类型的文件的情况下，仍进行COMMIT 操作（对该进程这前的事务进行确认处理）f

4. 关于日志的确认操作，在后面会另设专门章节讲述。

**COMP (Compare) 比较**

## 1. 基本语法：

将Factory 1 与Factory 2 进行比较。

当Factory 1 &gt; Factory 2 时，打开HI 指示器；

当Factory 1 = Factory 2 时，打开LO 指示器；

当Factory 1 &lt; Factory 2 时，打开EQ 指示器。

Factory 1 Operation Factory 2 Result HI LO EQ

FLD01 COMP FLD02 56 57 58

当FLD01=2, FLD02=1 时，\*IN56=' 1' , \*IN57=' 0' , \*IN58=' 0'

当FLD01=2, FLD02=2 时，\*IN56=' 0' , \*IN57=' 0' , \*IN58=' 1'

当FLD01=1, FLD02=2 时，\*IN56=' 0' , \*IN57=' 1' , \*IN58=' 0'

## 7.3.5.2 D--E开头

**DEFINE (Field Definition)**

根据已知的字段，来定义新字段，如下：

Factory 1 Operation Factory 2 Result HI LO EQ

\*LIKE DEFINE FLD01 FLD02

这句话的意思，就是说 象定义字段FLD01 一样，定义字段FLD02 的类型、长度” 这个字段FLD01，必须是个已定义的字段/变量。

**DELETE {(E)} (Delete Record)**

删除当前记录，语法如下：

Factory 1 Operation Factory 2 Result HI LO EQ

DELETE 文件记录格式名

这里，在做DELETE 操作前，必须先定位到具体的记录上（使用CHAIN、READ 等语句）；同时，文件在F 行必须使用修改的方式来声明。

当文件定位到对象记录时，对象记录会被锁定，此时，对象记录无法被其它程序修改；



如果执行了DELETE 操作，自然会解锁，不过别的程序也找不到对象记录了（因为被删除）实际使用中，通常并不会对文件中的记录进行物理删除，而是修改某个标识状态的字段的值，所以使用这条命令要慎重。

#### DIV {(H)} (Divide) 数学运算—除

DIV 操作码，表示数学运算的“除”，常与操作码MVR 一起来使用。

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
FLD01	DIV	FLD02	N			
	MVR		M			

上面两句话的意思，是说，用FLD01 来除FLD02，将商赋值到变量N 中，将余数，赋值到变量M 中。

(N, M 都是数字型变量)再具体一点，如果FLD01 = 10, FLD02 = 3，执行完这两句操作码之后  
N = 3 （10 / 3 的商）

M = 1 （10 / 3 的余数）

#### DO (Do) 循环

最常用的循环方法之一，适用于已知循环次数的循环，与ENDDO 搭配使用，每一个DO，必须要配合一个ENDDO。基本语法如下

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
1	DO	10				

循环中处理的内容

ENDDO

上面的意思，就是说固定循环10 次。

不过呢，在实际使用中，我们常常需要知道当前循环到了第几次处理，这里，可以：

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
1	DO	10	N			

处理过程

ENDDO

这个N，是一个整数型变量（小数位为0 的P 型吧），它所表示的就是循环的次数。

如第一次循环，N=1；第二次循环，N=2

所以， 1 DO 1，就表示只循环一次，而不是表示死循环。

#### DOU {(M | R)} (Do Until) 还是循环

也是循环，不过当满足Extend Factory 2 项的条件之后，结束循环。如下：

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
	DOU	FLD01>FLD02				

处理过程

ENDDO

上面这句话，就是说当FLD01 小于或等于FLD02 时，进行循环；当满足条件，即FLD01大于FLD02 时，结束循环。在RPGLE 的写法中，这个条件可以写得很复杂。当然，很复杂的时候，要记得



多用括号，以免得逻辑判断与设计不一致。

#### DOUxx (Do Until) 又是循环

这应该是RPG 的写法，上面的循环也可以写做：

```
FLD01 DOUGT FLD02  
ENDDO
```

不过，正如果前面所说的“ ANDxx” 操作码一样，RPGLE 的表示手法可以比RPG 更直观，更好维护，所以这里也是一样的原理，有了“ DOU” 之后，我们就可以不用“ DOUxx” 了。

#### DOW {(M | R)} (Do While) 循环，又见循环

这个循环的意思，与DOU 正好相反，它的意思是满足Extend Fatcory 2 项的条件时，才进行循环，不满足条件时，不循环

```
Factory 1      Operation      Factory 2      Result      HI LO EQ  
              DOW              FLD01>FLD02
```

处理过程

```
ENDDO
```

上面这句话，就是说当FLD01 小于或等于FLD02 时，不进行循环循环；当满足条件，即FLD01 大于FLD02 时，才进行循环。在RPGLE 的写法中，这个条件可以写得很复杂。当然，很复杂的时候，要记得多用括号，以免得逻辑判断与设计不一致。

注意：在实际使用过程中，我常常只使用DO, DOW 这两种循环。而且使用DOW 时，常常使其条件永远成立（即死循环，比如说1=1 就永远成立，那么 DOW 1=1 就是个死循环），然后在循环体中用IF 语句判断何时退出循环（使用LEAVE 语句），我认为这样会比较直观，而且很少会在逻辑上出错。

#### DOWxx (Do While) 最后一个循环了

跟 DOU 与DOUxx 的关系一样，有了DOW 之后，就不需要DOWxx 了，不多说了。

#### DSPLY {(E)} (Display Function) 屏幕显示

```
Factory 1      Operation      Factory 2      Result      HI LO EQ  
FLD01          DSPLY
```

就是在屏幕上，显示FLD01 的内容。FLD01 可以是字符、数字型变量。也可以直接就是字符、数字。

#### ELSE (Else) 逻辑判断语句

常见用法：

```
Factory 1 Operation Factory 2 Result HI LO EQ  
IF 条件判断  
处理内容一  
ELSE  
处理内容二  
ENDIF
```

不满足条件判断的时候，就执行ELSE 与ENDIF 之间的处理内容二；

满足条件判断时，执行IF 与ELSE 之间的处理内容一。

IF 与它最临近的ELSE、ENDIF 配对；

同理，ELSE 也是与它最临近的IF、ENDIF 配对。

注意多层IF 嵌套时的逻辑判断，这个要自己多尝试，不讲了。

ELSEIF {(M | R)} (Else/If) 逻辑判断语句

与ELSE 类似，不过将条件判断语句也写在了同一行。

ELSEIF 不需要相应的ENDIF 语句。即

IF 条件判断1

处理内容一

ELSEIF 条件判断2

处理内容二

ELSE

处理内容三

ENDIF

当程序满足条件判断1，将执行IF 与ELSEIF 之间的处理内容一；

当程序不满足条件判断1，满足条件判断2，执行ELSEIF 与ELSE 之间的处理内容二；

当程序连条件判断2 的内容都不满足时，执行ELSE 与ENDIF 之间的内容处理内容三。

也就是说，这时的ELSE，是和ELSEIF 进行配对的，要注意。

ENDyy (End a Structured Group)

ENDIF 对应 IF、IFxx

ENDSR 对应 BEGSR

ENDDO 对应 DO、DOW、DOU、DOUxx、DOWxx

ENDSC 对应 CASxx

ENDSL 对应 SELECT

ENDFOR 对应 FOR

ENDSR (End of Subroutine) 子过程结束语句

### EVAL {(H | M | R)} (Evaluation) 赋值语句

1. 基本语法：

Factory 1	Operation	Factory 2	Result	HI LO EQ
	EVAL	FLD01=FLD02		

赋值，使用FLD01 的值，等于FLD02 的值。FLD01 与FLD02 的类型必须相同（即同为字符型，或同为数字型，但长度可以不同。

2. 当FLD01、FLD02 同为字符型时，EVAL 语句其实等价于：

EVAL	FLD01=*BLANKS	
MOVE	FLD02	FLD01

即先清空FLD01（当FLD01 长度长于FLD02，且有初始值时，这个操作就有意义了），然后再将FLD02 左对齐赋值到FLD01 中。所谓左对齐的含义，就是说将FLD02 至左向右（含空字符），依次向FLD01 字段中赋值。当FLD02 长度大于FLD01 时，右边的字符在赋值时将会被截去。

3. 当FLD01、FLD02 同为数字型时，且FLD02 的值，超过FLD01 的长度时，EVAL语句会直接报错，程序异常中断；

4. 当FLD01、FLD02 同为字符型时，且FLD01 的长度大于FLD02 时，将会将FLD02的值左对齐赋值到FLD01 中，并且将后面内容全部清空。

比如FLD01=' 12345678'（8 位字符型），FLD02=' 54321'（5 位字符型）执行了 EVAL FLD01=FLD02 之后，FLD01=' 54321 '

5. 当FLD01、FLD02 同为字符型，且FLD01 的长度小于FLD02 时，会将FLD02 的内容左对齐，再赋值到FLD02 中。

6. 字符的拼接：

Factory 1 Operation Factory 2 Result HI LO EQ

EVAL FLD01=FLD02+FLD03+' AAA' +' BB' +FLD04

FLD01、FLD02、FLD03、FLD04 都必须是字符。FLD02、FLD03 如果字符未满，有空格，EVAL 操作码不会将空格去掉。如果太长，可以分多行写，“+”在上在下都可。即

EVAL FLD01 = FLD02 +

FLD03

等价于

EVAL FLD01 = FLD02

+ FLD03

7. 数学运算：

Factory 1 Operation Factory 2 Result HI LO EQ

EVAL FLD01 =(((LD02+FLD03) \* FLD04) / 2) - FLD05

加、减、乘、除都在了。注意多用括号。

如果运算公式太长，一行写不完，也可以分行写，运算符在上行、下行都可以。

8. 四舍五入：

对于数字运算，EVAL(H)，是四舍五入的运算，如。

EVAL(H) FLD01 = FLD02 / FLD03

而EVAL，仅仅是四舍，没有五入。

#### EXFMT {(E)} (Write/Then Read Format) 显示屏幕文件并等待输入

这个操作码的意思，是显示屏幕，并等待输入，语法为：

Factory 1 Operation Factory 2 Result HI LO EQ

EXFMT 记录格式名

当程序执行到这一句时，表示显示“记录格式名”所对应的子文件（由DSPF 定义，显示在屏幕上），同时等待用户进行输入操作。

当用户输入预定义的热键（F3，F4 等等）、或执行输入操作后(按下输入键)，程序才会继续向下执行。

#### EXSR (Invoke Subroutine) 执行子过程

Factory 1 Operation Factory 2 Result HI LO EQ

EXSR 子过程名

#### EXTRCT {(E)}(Extract Date/Time) 取日期、时间型变量中的具体内容

首先，必须要有一个日期或时间型变量，这个可以在D 行定义，然后使用该变量中有值，比如

D FLD01 S D

D FLD02 S T

C MOVE \*DATE FLD01

C TIME FLD02

这样，就有了一个日期型变量FLD01，一个时间型变量FLD02，且两个变量中都有值。再假设变量FLDYEAR 为四位字符型、FLDHOURL 为两位数字型

Factory 1 Operation Factory 2 Result HI LO EQ

EXTRCT FLD01:\*Y FLDYEAR

EXTRCT FLD02:\*H FLDHOURL

这时，FLDYEAR 表示当前的年份；FLDHOURL 表示当前的小时所以说，EXTRCT 操作码对应的对象变量，可以是字符型，也可以是数字型，但长度一定要与预期的对象长度相等（比如年份可以是字符，也可以是数字，但长度必须为4 位；月份必须为2 位，依此类推）而冒号后面所跟的参数，与ADDUR 操作码一样，如下：

对应于日期型变量：

\*D表示天，也可用\*DAY

\*M表示月，也可用\*MONTH

\*Y表示年，也可用\*YEAR

对应于时间型变量

\*H 表示小时，也可用\*HOUR

\*MN 表示分钟，也可用\*MINUTE

\*S 表示秒， 也可用\*SECOND

#### 7.3.5.3 F--N开头

##### FOR (For) 又是循环

FOR 操作码，也是一种指定次数的循环，与“ DO” 循环有异曲同工之妙。

Factory 1 Operation Factory 2 Result HI LO EQ

FOR N=1 TO 10

处理语句

ENDFOR

等价于

Factory 1 Operation Factory 2 Result HI LO EQ

1 DO 10 N

处理语句

ENDDO

都是循环10次，并且将当前循环次数赋值到数字型变量N中

### GOTO (Go To) 跳转语句

GOTO，跳转语句。跳转到指定的标签处，需要用“TAG”语句来定义，如：

Factory 1 Operation Factory 2 Result HI LO EQ

处理语句1

IF 条件判断

GOTO FHSTAG

ENDIF

处理语句2

FHSTAG TAG

在这里，执行完处理语句1时，如果“条件判断”成立，将会直接跳至“FHSTAG TAG”处，不执行处理语句2。主过程不能跳转至子过程；（主过程我用来表示是与子过程相对应的词，指C行顺序第一行，至程序结束语句之间的语句）子过程不能跳转至其它子过程，只能在本子过程内部跳转，或跳转到主过程；技术可以使用跳转语句来实现循环，不过不建议。

### IF {(M | R)} (If) 条件判断

条件判断语句，必须与ENDIF一起使用

```
Factory 1      Operation      Factory 2      Result      HI LO EQ
              IF              ((FLD01=FLD02) OR (FLD01>FLD03)) AND
                                   (FLD03=FLD04)
```

处理语句

ENDIF

当条件判断成立时，才会执行下面的处理语句。

条件判断可以使用括号，OR，AND，来表达复杂的逻辑关系

IF 必须有对应的ENDIF，所以建议写程序时，写无IF就加个ENDIF，然后再来写中间的处理语句，以免遗漏。

IF 语句，还可以这种用法：

IF \*IN(20)

ENDIF

这时，等价于

IF \*IN20='1'

ENDIF

### IFxx (If) 条件判断

与ANDxx类似，这是RPG的语法，一行代码只能表示一个条件，如

```

Factory 1  Operation  Factory 2  Result  HI  LO  EQ
FLD01      I FEQ      FLD02
FLD01      ORGT      FLD03
处理语句

```

ENDIF

与AND 和ANDxx 类似，由于IF 语句在表达复杂的逻辑关系时更直观，所以通常我只使用IF，而不是IFxx 了。

#### ITER (Iterate) 循环中，不处理此语句之后的语句

```

Factory 1 Operation Factory 2 Result HI  LO  EQ
1 DO 10 N
处理语句1
IF 条件判断
ITER
ENDIF
处理语句2
ENDDO
等价于

```

```

Factory 1 Operation Factory 2 Result HI  LO  EQ
1 DO 10 N
ITERTAG TAG
处理语句1
IF 条件判断
GOTO ITERTAG
ENDIF
处理语句2
ENDDO

```

不知道这个例子能不能让人明白，都是在说当条件判断成立时，就不执行处理语句2，直接从循环头开始处理。ITER 语句执行后，程序相当于跳转至循环体中的第一句，重新向下运行（即执行循环）。当有多层循环嵌套时，ITER 仅作用于最里层的循环。当循环中执行了子过程时，子过程中不能对该循环进行ITER 操作。也就是说：DO，ENDO，ITER 这些语句必须在一块（即要么都在主过程中，要么都在同一个子过程中）

#### KFLD (Define Parts of a Key) 定义组合键值中的字段

这个其实应该与KLIST 一起来讲，因为这两个操作是在一起使用的。一个PF 对应的逻辑文件LF，可能会有多个KEY 值，那么，我们要根据多个KEY 值，对于该逻辑文件进行查询定位操作时，就需要使用KLIST、KFLD 这两个操作码：

```

Factory 1      Operation      Factory 2      Result      HI  LO  EQ
FHSKEY        KLIST
//定义组合键值FHSKEY

```

```
KFLD          FLD01          //组合键的首个字段为FLD01
KFLD          FLD02          //组合键的第二个字段FLD02
```

组合键值的使用方式，就是将组合键视为一个变量，进行各种操作，如：

```
FHSKEY      CHAIN      FMTFHS
```

或

```
FHSKEY      SETLL      FMTFHS
```

意思，就是说按组合键值，查询定位到逻辑文件PFFHSL3 中去。

这里要注意，PFFHSL3 的键值是FHS01、FHS02（即PF 文件中的字段），而程序中的组合键值为FLD01、FLD02 两个程序中定义的变量，只要类型、长度与键值FHS01、FHS02相等即可。同时，对文件进行查询定位操作后，在没有对FLD01、FLD02 赋值之前，这两个字段的值是不会发生变化的（即不会更改组合键的键值）；但是，组合键中的字段，也可以直接定义为PF 文件中的字段，如

```
Factory 1      Operation      Factory 2      Result      HI  LO  EQ
FHSKEY          KLIST
                KFLD          FHS01
                KFLD          FHS02
```

此时，可以将PF 文件中的字段FHS01、FHS02 先视为两个普通的变量，进行赋值，然后用组合键FHSKEY 对PFFHSL3 进行查询定位；当执行完CHAIN，或READ 语句后，因为读取到了文件中的数据，所以FHS01、FHS02 的值将发生变化，也即是组合键FHSKEY的键值发生了变化。基于此，所以通常我不会将组合键的字段设置为查询文件中的字段，而是定义为几个独立的临时字段，以免需要过于频繁的考虑键值的更改。

**KLIST (Define a Composite Key) 以上面的KFLD 操作码**

**LEAVE (Leave a Do Group) 退出当前循环**

退出当前循环，语法和注意事项基本与“ ITER” 相同：

```
Factory 1 Operation Factory 2 Result HI  LO  EQ
```

```
1          DO          10          N
```

处理语句1

IF 条件判断

LEAVE

ENDIF

处理语句2

ENDDO

等价于

```
Factory 1 Operation Factory 2 Result HI  LO  EQ
```

```
1          DO          10          N
```

处理语句1

```
IF 条件判断
GOTO LEAVETAG
ENDIF
处理语句2
ENDDO
LEAVE TAG
```

这两个例子都是在说当条件判断成立时，就不执行剩余的循环体，直接跳出循环。LEAVE 语句执行后，程序相当于跳转至ENDDO 循环结束处，然后向下执行ENDDO之后的循环外的执行语句。

当有多层循环嵌套时，LEAVE 仅作用于最里层的循环。

当循环中执行了子过程时，子过程中不能对该循环进行LEAVE 操作。也就是说：DO，ENDO，LEAVE 这些语句必须在一块（即要么都在主过程中，要么都在同一个子过程中）

### LOOKUP (Look Up a Table or Array Element) 对数组的查询定位

假设已定义数组DIMDATA，每条记录四位长，共三条记录的数组，且数组中已赋值如下：

```
DIMDATA(1) = ' 0001'
DIMDATA(2) = ' 0002'
DIMDATA(3) = ' 0003'
```

那么，下面的语句就是对数组的查询定位操作：

```
Factory 1 Operation Factory 2 Result HI LO EQ
EVAL N=1
'0002' LOOKUP DIMDATA(N) 54
```

首句N=1，表示从第一条记录开始查询；如果N=2，即表示从第2 条记录开始查询；如果N 大于数组定义时的总记录数（如N=4），则下面的LOOKUP 语句会报错，跳出程序；

第二句，表示查询 数组DIMDATA 中，内容为' 0002' 的，是第几条记录；其中，54 为EQ 指示器。当找到记录时，打开指示器，\*IN54 = ' 1' ；当未找到记录时， 指示器处于关闭状态，\*IN54=' 0' 。与操作码CHINA 的指示器含义刚好相反；

在这个例子中，执行完LOOKUP 语后，\*IN54=' 1' ， N = 2（即数组DIMDATA 中，有内容为' 0002' 的记录，是第二条记录。当数组中有多条内容相同的记录时，N 为顺序查询到的第一条记录当数组中无相应记录时，\*IN54=' 0' ， N = 初始值，此例中，即N = 1

### MOVE (Move) 赋值语句

#### 1. 基本语法：

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
	MOVE	FLD01	FLD02			

FLD01、FLD02 可以是不同的类型。即允许一个为字符，一个为数字

#### 2. 允许在MOVE 操作中，对Result 字符进行定义。

#### 3. MOVE 操作码，使用右对齐赋值。所谓右对齐，即是将Factory 2 的最右边一个字符，赋值



到Result 的最右边的一个字符；然后再将Factory 2 向左取一位字符，赋值到Result 从最右边起，左移一位的字符是，依此类推。直到取完Factory 2 的值，或Result 已赋满值。

4. 由上述其实已可看出，MOVE 操作码在对字符赋值时，使用的是覆盖方式赋值。即如果FLD02 将在赋值的字符处，原来有值，将会将原来的值覆盖；如果已有值时，未执行赋值操作，将仍然保留原值。

5. 当FLD01、FLD02 都为字符型：如果FLD01 的长度为5 位，FLD02 的长度为8 位。FLD01 的内容为' 12345' ，FLD02 的内容为' ABCDEFGH' ，当执行了MOVE 操作后，FLD02 的内容为' ABC12345' ，可以看到，后5 位由' DEFGH' 变成了FLD01 的' 12345' ，而前三位仍然保留原来的值' ABC' ；

要注意的是，在字符型变量中，空字符也会参与赋值。如果FLD01 的内容为' 12 '，即后三位是空时，那么执行MOVE 操作后，FLD02 的值将会是' ABC12 '。看到了没有，最后三位也与FLD01 一样，是空；而前三位FLD01 的值保持不变，仍为' ABC' 。反之，当FLD01 的长度为8 位，' 12345678' ，而FLD02 的长度为5 位' ABCDE' 时，执行完MOVE 操作后，FLD02 的值为' 45678' ，即从最右边开始，向左赋值；MOVE 操作表示右对齐，空也算字符。

6. 数字赋值到字符的处理：

当FLD01 为数字(5, 2)，FLD02 为字符(8)时，MOVE 操作将会把数字的小数点去掉，将数字转换为字符，右对齐赋值到字符变量中。数字型变量如果未满足，前面都视为空，而不是0；如果FLD01 为123.45，FLD02 为' ABCDEFGH'，赋值之后，FLD02=' ABC12345'，前三位保持FLD02 原有的值。如果FLD01=123.45，FLD02 原来为空，赋值之后，FLD02=' 12345'；（字符型变量前三位保持原来的空值）如果FLD01=3.45，FLD02 原来为空，赋值之后，FLD01=' 345'。（数字型变量未满足，前面也视为空，而不是0）如果数字的长度大于字符的长度，系统将会自动从左边开始，将大于的部分截去。如FLD01 长度为10, 2，等于12345678.90 时，赋值后，FLD02 会等于' 34567890'

7. 字符赋值到数字的处理：

当FLD01 为字符(8)，FLD02 为数字(5, 2)时，MOVE 操作会将字符右对齐赋值到数字中，如果字符后面有空，系统会自动做补零处理；如果FLD01 为' 12345678'，FLD02 为0，赋值之后，FLD02=456.78；如果FLD01 为' 12345 '，即后三位为空时，赋值之后，FLD02=450.00，自动补0；当数字长度大于字符的长度，如FLD02 为(10, 2)时，数字最高位在赋值时，将会保持原值，如FLD02=12345678.90，FLD01=' 87654321'，赋值之后，FLD02=12876543.21，注意最高位的12 保持未变所以通常，字符转数字时，长度最好保持一致，以免得赋值时有意外。

8. 日期、时间型变量与字符、数字的转换

日期型变量与字符的转换，会带上分隔符。如果日期型变量为' 2007-02-12' 时，字符型变量也需要定义为8+2 位，当执行完赋值操作时，字符型变量即为' 2007-02-12'，注意，是有分隔符的；时间型变量也一样，只是长度需要6+2位。反之，当字符型变量赋值到日期型变量中，也是需要这个分隔符的；

日期型变量与数字的转换，将会自动去掉分隔符（非常好吧），如果日期型变量为' 2007-02-12' 时，赋值到一个8 位的数字型变量，该变量就等于20070212；反之，数字赋值到

日期型变量中时，也是需要8 位纯数字；时间型变量也一样处理，只要长度需要为6 位。

### MOVEA {(P)} (Move Array) 数组赋值

该操作码仅作用于数组。

假设已定义数组DIMDATA，每条记录二位长，共三条记录的数组，且数组中已赋值如下：

DIMDATA(1) = ' 01'

\_\_ DIMDATA(2) = ' 02'

DIMDATA(3) = ' 03'

当执行完MOVEA 操作后：

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
	MOVEA	*BLANK	DIMDATA			

DIMDATA 中所有的要素都变成了空值。

MOVEA 还可以将一个变量拆分赋值到数组中，仍是上例，假设有一个6 位字符型变量 FLD01，其值为' ABCDEF'，那么执行语句：

MOVEA	FLD01	DIMDATA
-------	-------	---------

后，数组中的值将是如下：

DIMDATA(1) = ' AB'

DIMDATA(2) = ' CD'

DIMDATA(3) = ' EF'

当变量的长度，小于（数组单条记录长度\*总记录数）时，超出部分的数组的值，保持原值。

仍是上例，假设FLD01 为4 位长的字符变量，值为' ABCD' 时，执行MOVEA 操作后，数组中的值如下：

DIMDATA(1) = ' AB'

DIMDATA(2) = ' CD'

DIMDATA(3) = ' 03'

注意，最后一个记录DIMDATA(3)的值仍保持原' 03' 不变，没有被清空。

### MOVEL {(P)} (Move Left) 左对齐赋值语法

与MOVE 操作码相同，所不同的是对齐方式为左对齐，即是将Factory 2 的最左边一个字符，赋值到Result 的最左边的一个字符；然后再将Factory 2 向右取一位字符，赋值到Result 从最左边起，右移一位的字符，依此类推。直到取完Factory 2 的值，或Result 已赋满值。

在进行长度不同的字符之间赋值时，可按照上面这个原则，根据MOVE 操作码中的案例，想想不同情况的结果，这里不做一一列举。

要注意的是，当使用MOVEL，由字符转为数字时，后面的空格会自动补0，如' 123'（后面5 位都为空），转到数字型变量中，就变成了123000.00。因为大部分情况下，我们定义的标准，都是字符型左对齐，数字型右对齐。所以此时字符转数字要注意，以免无端扩大倍数。

### MULT {(H)} (Multiply) 数学运算—乘

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
FLD01	MULT	FLD02	FLD03			

等于

EVAL FLD03 = FLD01 \* FLD02

可以对当前行对FLD03 进行定义；

Factory 1 可以不填写；

必须保证FLD03 的位数足够大，否则超长时，程序会报错异常退出。

### MVR (Move Remainder) 数学运算—除法运算取余

这个操作码，在讲操作码DIV 时，已讲过：

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
FLD01	DIV	FLD02	FLD03			
	MVR		FLD04			

当FLD01=11，FLD02=4，

FLD03=2 （商）

FLD04=3 （余）

#### 7.3.5.4 0--R开头

### OPEN {(E)} (Open File for Processing) 打开文件

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
	OPEN	文件名				

OPEN 后面的对象，必须是在当前程序中已声明的文件名（不是文件的记录格式名），而且在OPEN 操作之后，在程序结束之前，必须有对应的CLOSE 操作。使用OPEN 操作，文件在声明时，必须使用USROPN 关键字（详见D 行说明）。

### ORxx (Or) 逻辑判断—或

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
FLD01	IFGT	FLD03				
FLD01	OREQ	FLD02				

等价于

IF FLD01>FLD03 OR FLD01=FLD02

与IF、IFxx，AND、ANDxx 类似，RPGLE 的写法OR，比RPG 的写法ORxx 要灵活，

而且可以用来表达一些复杂的逻辑关系。有鉴于此，所以通常IF 语句中，我会以OR 为主，基本不用ORxx。如果在编程序方面，公司/项目组无硬性要求，那我觉得还是少用ORxx 吧，总觉得这种写法的逻辑关系看起来不直接，尤其是有很复杂的AND, OR 时。

### OTHER (Otherwise Select) 分支语句的判断

与分支语句SELECT 一起使用，表示不符合上述所有条件时的操作，如下：

Factory 1 Operation Factory 2 Result HI LO EQ

```
SELECT
WHEN 条件判断1
    处理语句1
WHEN 条件判断2
    处理语句2
OTHER
    处理语句3
ENDSL
```

在这个例子中，当满足条件判断1 时，运行处理语句1，运行结束后跳至ENDSL 处；如果不满足条件判断1，则程序继续向下执行，判断是否满足条件判断2。

当满足条件判断2 时，运行处理语句2，跳至ENDSL；当不满足

当不满足条件判断2 时，程序继续向下执行，当读到OTHER 操作码时，无条件运行处理语句3（即当程序当前不满足以上所有条件判断时，则执行OTHER 之后的语句。

处理语句允许有很多句；条件判断可以写得很复杂，也允许对不同的字段进行判断；比如说C 语言也有分支语句switch，但是这个语句只能对一个字段进行分支判断，ILE 语言与它不同，允许对不同的字段进行判断就我目前掌握的测试情况，上述的SELECT—WHEN—OTHER—ENDSL，其实也可以写做：

```
IF 条件判断1
    处理语句1
ELSEIF 条件判断2
    处理语句2
ELSE
    处理语句3
ENDIF
```

即WHEN 与ELSEIF 是类似的，这样说，应该可以明白了吧。

总之，SELECT—ENDSL 是一个很好用的语法，尤其是在表示很多不同的分支处理时。

### READ {(N | E)} (Read a Record) 读取记录

1. 基本语法：

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
	READ	文件记录格式名		45	46	

READ后面跟的，必须是声明的文件记录格式名；

LO 指示器表示锁表指示器，当在指定的时间（CHGPF，WAITRCD 项可看到），需要读取的记录仍被锁，将会打开LO 指示器，即\*IN45=' 1' ；

EQ指示器为是否读到指示器。当未读到任何记录时，打开EQ 指示器，即\*IN46=' 1'

2. 当文件在程序中，是用只读的方式声明时，READ 操作并不会造成锁表；

如果文件在程序中是用修改的方式声明，READ 操作成功后，该记录被锁；直到执行解锁操作（UNLOCK，或UPDATE），或READ 该文件的其它记录，才会解锁

如果文件是用修改的方式声明，但希望READ 操作不锁表时，那么就用READ（N），即

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
	READ(N)	文件记录格式名		45	46	

这样读文件，就不会锁记录，但是同时也不能修改记录。如果需要修改记录，那么在修改之前（包括对文件字段赋值之前），还必须再对该记录进行一次定位操作（比如CHAIN、READ 语句均可）。也就是说，如果要修改记录，必须先锁住当前记录。

3. 当执行READ 操作时，程序是根据游标当前在文件中所指向的位置，顺序读取下一条记录。

4. 执行READ 操作时，允许声明的文件没有键值。（即PF 文件）

READC {(E)} (Read Next Changed Record)

读下一次修改过的记录

READE {(N | E)} (Read Equal Key) 读取键值相等的记录

语法与READ 操作码大致一样，这里不再重复，只说不同的：

假设程序中已声明逻辑文件PFFHSL3（键值为FHS01+FHS02）

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
FHSKEY	KLIST					
	KFLD	FLD01				
	KFLD	FLD02				
FHSKEY	SETLL	FMTFHS				
	DOW	1=1				
FHSKEY	READE	FMTFHS			15	
	IF	*IN15=' 1'				
	LEAVE					
	ENDIF					
	ENDDO					

这段话的意思，就是定义组合键值FHSKEY，然后根据这个FHSKEY 在逻辑文件PFFHSL3 中去定位，循环读取PFFHSL3 中，FHS01、FHS03 与FLD01、FLD02 相等的记录。

当读取记录结束，或键值不等时，退出循环（\*IN15 是EQ 指示器）。如果将READE 操作码换成READ 操作码的话（当然，Factory 1 处也就不能有值），就没有“键值不等时退出循环”这一层意思，只是读不到记录时就退出循环，但有时我们使用逻辑文件，仅仅是需要它的排序，而不需要读不到键值相等的记录就退出循环。

所以说，使用READ 操作码，还是READE 操作码，需要根据实际的要求来决定。以上的Factory 1 处填写值的系统处理，当READE 操作码在Factory 1 处未填写值时，系统实际上是将当前的值与读到的上一条记录的关键字进行比较，而不是与SETLL 时的键值做比较（读第一条记录不

做比较！），如果键值不等时，置EQ 指示器为1。也就是说，如果没有与FHSKEY 键值相同的录，那么系统并不是直接找开EQ 指示器，而是会一直保持正常地往下读，直到找到与读到的第一条记录关键字不同的记录，才会打开EQ 指示器，所以要注意。

#### READP {(N | E)} (Read Prior Record) 读取记录—光标上移

简单来说，READ、READE 操作时，光标在数据文件中，是下移的；即读完第一条记录，光标指向第二条记录；读完第二条记录，光标指向第三条记录，依此类推，直至最后一条记录。但READP 则正好相反，光标是上移的，即读完第三条记录后，光标指向第二条记录；读完第二条记录后，光标指向第一条记录，直至读完第一条记录。一般来说，用READ、READE 的概率会比READP、READPE 的概率高得多，不过在某些情况下，使用READP 操作，又的确会很省事，大家可在编程时多实践。

#### READPE {(N | E)} (Read Prior Equal)

是指光标上移，按键值去读取文件。与READP 的关系，就类似于READE 与READ 的关系。

#### RESET {(E)} (Reset)

将数据结构赋值成为初始值。

注意是初始值，不是清空。

如定义结构：

D FHSDS DS

D FHS01 10 INZ (' ABCD' )

D FHS02 5 INZ (' EFGH' )

那么，不管对该结构如何赋值，当执行语句：

C RESET FHSDS

之后，FHS01 将会变成' ABCD，FHS02 将会变成' EFGH' ，即恢复成为初始值。

#### RETURN {(H | M | R)} (Return to Caller)

RETURN 是程序结束。

在前面，“简单的程序流程”中，我们讲过，“SETON LR” 与RETURN 这两句话一起，做为程序的结束。这里，再详细解释一下两者之间的区别，以及关系：如果不写RETURN，只写 SETON LR”，程序执行完最后一句之后，将会再从第一句开始执行，造成死循环。

在简单的程序流程这个例子中，程序原来只想修改读到的第一条记录，而如果没有RETURN 的话，将会把所有的记录都修改掉，直到最后找不到可修改的记录，然后系统报错，异常中断。如果只写RETURN，不打开指示器\*INLR，根据blogliou 所说“程序不会强制将内存中的数据写到磁盘中。400 缺省的是BLOCK 输出，即数据记录满一个BLOCK 块时才会将这一组记录写到磁盘中。那么如果这时BLOCK 没满，数据信息不会立刻写到磁盘中。之后有其它作业用到该文件，读取的数据就不完整。”但如果文件有唯一键字，或记录日志，必须同步写时，其实BLOCK 实际被忽略，也就是此时不会有错。

### ROLBK {(E)} (Roll Back)

#### 1. 基本语法

Factory 1 Operation Factory 2 Result

ROLBK

2. 该操作码无其它参数，就是指对事务处理进行回滚操作。

3. ILE 程序中，ROLBK 操作可随时进行，也允许在没有声明COMMIT 类型的文件的情况下，仍进行ROLBK 操作（对该进程这前的事务进行确认处理）

### 7.3.5.5 S--Z开头

#### SCAN {(E)} (Scan Character String) 扫描字符串

扫描字符或字符串Factory 1 在对象字符串Factory 2 中是否存在

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
FLD01	SCAN	FLD02	N			26

FLD01 可以是字符，也可以是字符变量；可以是一位长，也可以是多位长。

当FLD01 在FLD02 中存在时，EQ 指示器打开，即\*IN26=' 1' ，同时将FLD02 中的起始位置，赋值给N；

当FLD01 在FLD02 中不存在时，EQ 指示器保持关闭状态，即\*IN26=' 0' ，同时N=0允许从FLD02 中的指定位置开始检查：

FLD01 SCAN FLD02: 2 N 26

如上句，即表示从FLD02 的第2 位，开始扫描。在实际使用中，比如说我们判断某个字符是否为数字，就可以先定义一个0—9 的常量，然后将要判断的字符去SCAN 一下这个常量。

#### SELECT (Begin a Select Group) 分支语句

在操作码“ OTHER” 中讲过，为方便读者，列出简单语法如下：

Factory 1 Operation Factory 2 Result HI LO EQ

SELECT

WHEN 条件判断1

处理语句1

WHEN 条件判断2

处理语句2

OTHER

处理语句3

ENDSL

要注意，SELECT 操作码，必须有对应的ENDSL 操作码，否则编译无法通过。

#### SETGT {(E)} (Set Greater Than) 定位操作—大于

举个例子吧，假设文件中有一个字段，是标识顺序号的，1、2、3、4。即该字段为1，表示第一条记录，该字段为2，表示第2 条记录。那么：



Factory 1	Operation	Factory 2	Result	HI	LO	EQ
2	SETGT	文件记录格式名				
	READ	文件记录格式名				

这个READ 操作，READ 到的，是第3 条记录。也就是说，SETGT 操作码，会将游标定位到大于键值的第一条记录前。在实际使用中，如果我们是按逻辑文件读取，而且读了一条记录之后，对其键值相同的记录都不需要再读取时，就可以用SETGT，不过需要注意，Factory 1 项，需要是与键值相同的变量，即如果文件是使用多个字段做为键值，那么我們也需要先定义一个组合键值的变量，然后Factory 1 处填写这个组合键值的变量名。

当声明文件的键值有多项时，Factory 1 项的键值，允许小于文件的键值，但顺序必须一致。即声明的文件如果键值为：FHS01、FHS02、FHS03，那么我们在程序中定义三个类型与之相同的变量FLD01、FLD02、FLD03，以下写法都是有效的

```
FLDKEY KLIST
      KFLD   FLD01
      KFLD   FLD02
      KFLD   FLD03
FLDKEY SETGT 文件记录格式名
FLDKEY KLIST
      KFLD   FLD01
      KFLD   FLD02
FLDKEY SETGT 文件记录格式名
FLD01  SETLL 文件记录格式名
```

### SETLL {(E)} (Set Lower Limit) 定位操作—小于语法

与SETGT 相同，含义与SETGT 不同。SETLL 操作码，会将游标定位到与键值相等的第一条记录之前，仍是上例，如果是

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
2	SETLL	文件记录格式名				
	READ	文件记录格式名				

那么READ 操作码读到的记录，就是第2 条记录，看到了吧，和SETGT 不同。

SETLL 操作码还可以用来简单判断当前键值是否存在有记录，以PFFHSL3 为例（键值为FHS01、FHS02）

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
FHSKEY	KLIST					
	KFLD	FLD01				
	KFLD	FLD02				
	EVAL	FLD01=' 01'				
	EVAL	FLD02=' 02'				



FHSKEY      SETLL 文件记录格式名 44

当文件中有相应记录时，EQ 指示器打开，即\*IN44=' 1'

当文件中无相应记录时，EQ 指示器关闭，即\*IN44=' 0' （与CHAIN 正好相反，要注意）而在这种用法中，SETLL 与CHAIN 的区别在于，CHAIN 是定位读取了记录，而SETLL仅仅只是判断该记录是否存在。所以用SETLL 操作，不能修改记录，也无法取出记录的值。只能判断记录是否存在。如果要修改记录，或取出记录的值，还需要有一个读取定位的操作，如READ，或READE、READP 等（最常用的，应该就是READ 操作）

#### SETOFF (Set Indicator Off) 关闭指示器

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
	SETOFF			10	11	12

等价于

EVAL	*IN10=' 0'
EVAL	*IN11=' 0'
EVAL	*IN12=' 0'

在SETOFF 这个操作码中，指示器填在HI、LO、EQ 哪里都没关系，都是表示要被关闭的指示器

#### SETON (Set Indicator On) 打开指示器

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
	SETON			10	11	12

等价于

EVAL	*IN10=' 1'
EVAL	*IN11=' 1'
EVAL	*IN12=' 1'

在SETON 这个操作码中，指示器填在HI、LO、EQ 哪里都没关系，都是表示要被打开的指示器

#### SORTA (Sort an Array)

数组排序

#### SQRT {(H)} (Square Root) 开方

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
9	SQRT	3	N			

这时，N=3（因为3 的平方为9）

9、3 都可以是数字型变量，或者直接是数字

SUB {(H)} (Subtract) 减法操作

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
FLD01	SUB	FLD02	FLD03			
	SUB	FLD02	FLD03			

### SUBDUR {(E)} (Subtract Duration) 日期相减

#### 1. 减日期

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
FLD01	SUBDUR	N: *Y	FLD02			

表示将日期型变量FLD01 减去N 年，赋值到日期型变量FLD02 中；

N 可以是一个数字型变量，也可以就是一个数字，N 允许为负数

\*Y, \*M, \*D（还有其它的参数值，可见ADDUR，其中有详细解释）

#### 2. 判断两个日期型变量之间的天/月/年数

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
FLD01	SUBDUR	FLD02	N: *D			

这时，N 做为一个结果变量，表示日期型变量FLD01 与FLD02 之间的天数

### SUBST {(P | E)} (Substring) 取字符/字符串

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
2	SUBST	FLD01: 3	FLD02			

表示从字段FLD01 的第3 位开始，取2 位，左对齐赋值到字段FLD02 中。

要求字段FLD01 的长度必须大于或等于3+2 位，否则程序会报错。

可以尝试用%SUBST 语句，也是等价的，如下

```
EVAL FLD02=%SUBST(FLD01: 3: 2)
```

表示的是同样的意思。

起始位数3，取的长度2，在两种写法之下，都可以使用数字型变量来表达。

相比较之下，%SUBST 还有一种用法，就是对字符的指定位置赋值，这个就厉害了：

```
EVAL %SUBST(FLD02: 3: 2)=' 01'
```

看到了吧，这句话就是说，使字段FLD02 的第3、4 位（即从第三位开始，两位长）等于“ 01”

### TAG (Tag) 定义标签，与GOTO 同用

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
FHSTAG	TAG					

### TESTN (Test Numeric)

检查字段是否全是数字

### TESTZ (Test Zone)

测试区位

### TIME (Time of Day) --取当前系统时间

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
	TIME	FLD01				

FLD01 可以是时间型或数字型变量

### UNLOCK {(E)} (Unlock a Data Area or Release a Record) 解锁

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
	UNLOCK	文件记录格式名				

UNLOCK 是解锁操作，在某种程度上，可以将UNLOCK 视为ROLBK，将UPDATE视为COMMIT。即如果锁定某条记录，并对其字段进行赋值之后，使用UPDATE 语句，将会把修改后的结果保存下来，即修改文件，而UNLOCK 语句则不会修改文件，即否认了之前对文件字段做的赋值修改。

从程序的执行效率上来讲，UNLOCK 的执行效率是高于UPDATE 的，因为UPDATE操作时，系统需要对文件的每一个字段进行确认处理(DEBUG 时可以看到)，而UNLOCK 就是简单的解锁而已。

#### UPDATE (Modify Existing Record) 修改记录

这里需要说明一下，在执行UPDATE 的时候，必须先使用READ、CHAIN 等操作码锁定一条记录。如果未锁住记录，UPDATE 操作码将会报错。当执行了UNLOCK、UPDATE、以及ROLBK 语句时，等于是解锁，此时再执行UPDATE 操作码之前，必须再次锁住记录操作；

#### WHEN {(M | R)} (When) 分支判断语句中的条件判断

在操作码“ OTHER” ，“ SELECT” 中都讲过，仍列出简单语法如下：

Factory 1 Operation Factory 2 Result HI LO EQ

```
SELECT
WHEN 条件判断1
处理语句1
WHEN 条件判断2
处理语句2
OTHER
处理语句3
ENDSL
```

#### WHENxx (When True Then Select)

上面的语法，是RPGLE 的语法，WHENxx 是RPG 的语法，也就是

SELECT

FLD01 WHENEQ FLD02

处理语句1

.....

这样的语法，在表达复杂的逻辑关系时，必须与ANDxx，ORxx 一起使用，所以我不使用WHENxx 这个操作码。

#### WRITE (Create New Records) 写记录

常用的方式：

Factory 1	Operation	Factory 2	Result HI LO EQ
	CLEAR	文件记录格式名	
	EVAL	文件字段1=xxxx	
	EVAL	文件字段2=xxxx	
	WRITE	文件记录格式名	

表示在文件中写入一条新记录。文件需要声明为可写的。

通常会在给文件字段赋值之前，作一次CLEAR 操作来进行初始化，以避免不必要的麻烦。

#### XF00T {(H)} (Sum the Elements of an Array)

对数组字段的累加统计。

假设DIMDATA 定义为一个数字型的数组变量，FHS01 为一个足够大的数字型变量

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
	XF00T	DIMDATA	FHS01			

就表示将数组DIMDATA 中的所有记录的值都取出来，汇总相加，赋值到数字变量FHS01 中

#### XLATE {(P | E)} (Translate)

将一个字符串中指定的字符，更换成另外的字符。

举例：如MYCHAR1， MYCHAR2 都是两个20 位长的字符型变量

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
	MOVEL	'ABCAAAC123'	MYCHAR1			
'A': '9'	XLATE	MYCHAR1	MYCHAR2			

执行过这个语句之后，MYCHAR2 就等于“ 9BC999C123’”，即将字符串MYCHAR1 中所有的“ A” 都变成了“ 9”；XLATE 也可能指定起始位置。如上句更改为：

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
'A': '9'	XLATE	MYCHAR1: 4	MYCHAR2			

则MYCHAR2 等于“ ABC999C123”，指从第4 位开始（含第4 位），将“ A” 变成“ 9” 赋值。

#### Z-ADD {(H)} (Zero and Add) 向数字型变量赋值

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
	Z-ADD	FLD01	FLD02			

将数字型变量FLD01，赋值到数字型变量FLD02 中。

Z-ADD、MOVE 虽然同是赋值操作码，但Z-ADD 的用法就远没有MOVE 那么变化多

端，只能在数字型变量之间赋值。所以也没有什么可说的了。zero

如果要对数字型变量赋初值，使用\*ZERO

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
	Z-ADD	*ZERO	FLD02			

#### Z-SUB {(H)} (Zero and Subtract) 用0 减

Factory 1	Operation	Factory 2	Result	HI	LO	EQ
Z-SUB	FLD01	FLD02				
等价于						
0	SUB	FLD01	FLD02			
等价于						
	EVAL	FLD02=FLD01*(-1)				

#### \*ALL

\*ALL 是个很有意义的变量，举例：

EVAL FLD01=\*ALL' 0'

表示将字符型变量FLD01 赋值为全' 0'

而CLOSE \*ALL就表示关闭所有文件，意义与上面是不同的

### %LEN

取字符串的长度，举例：

(MYLEN 为数字型变量，FLD01 为字符型变量)

```
EVAL MYLEN = %LEN(FLD01)
```

这句话的意思，是指取字符串FLD01 的长度，不过通常这样用是没意义的，因为直接用%LEN 操作码，取到的是字符串的总长度，不是有效字符的长度，也就是说FLD01 长度为2，那么MYLEN 就恒等于2，不会变。就算变量FLD01 中没有值，取出的MYLEN 也等于2。所以，%LEN 通常会与%TRIM 或是%TRIMR 一起使用，语法在下面介绍。

### %TRIM, %TRIMR

都是去字符串变量中的空字符意思，%TRIM 是去字符串左边的空字符；%TRIMR 是去字符串右边的空格。通常我们在写程序中，都是默认字符串变量左对齐，所以我们使用%TRIMR 操作码的概率应该高一点。举例：

```
EVAL MYLEN = %LEN(%TRIMR(FLD01))
```

这时的MYLEN,就是指变量FLD01 中的有效长度(前提条件是FLD01 中如果有值,是左对齐)。如果FLD01 为空,那么MYLEN 为0;如果FLD01 首位有值,第二位为空,那么MYLEN 为1;如果FLD01 两位都不为空,那么MYLEN 就等于2。如果字符串左对齐,那么就使用%TRIMR还有一种用法,假设字符串FLD04 为4 位长的字符,FLD01, FLD02 都是2 位长的字符,且FLD01 等于“ A ”, FLD02 等于“ BC ”, 那我们执行：

```
EVAL FLD04 = FLD01 + FLD01 + FLD02
```

FLD04 就等于“ A A ”, 也就是第二位与第四位都是空的,最后加的FLD02 其实无效。而如果执行

```
EVAL FLD04 = %TRIMR(FLD01) + %TRIMR(FLD01) + FLD02
```

则FLD04 就等于“ AABC ”, 也就是说, 在这里, %TRIMR(FLD01), 是等价于单字符“ A ” 的

### MONITOR

监控程序信息。据说是可以屏蔽掉出错信息，避免程序异常中断，未经测试。

## 第八章 编辑错误处理与 DEBUG 调试

### 8.1 编辑错误处理

程序错误分为语法错误和逻辑错误。在源程序编译成对应的对象时，编译器会自动提示语法错误，并把错误信息写到相应的编译清单，存放在对应的作业队列中。通过查看作业队列中的编译清单，可以快速找到语法错误发生的原因和错误位置。

#### 8.1.1 编译清单中的结构化操作缩进

如果源程序中包括结构化操作码（例如 DO—END 或 IF—ESLE—END），你可能希望在源程序清单中这些操作有缩进显示。INDENT 可以让你指定是否显示缩进，并指定标记缩进的字符。在编译程序时，找到 INDENT 参数（默认值\*NONE）。如果想要缩进的话，那么最多指定两个字符来标记缩进。

例如，要指定让结构化操作缩进，并以显示(|)和空格标记，应指定 INDENT(' |')。

Create RPG/400 Program (CRTRPGPGM)

Type choices, press Enter.

Program . . . . .	PGM	>	<u>TEST01R</u>
Library . . . . .		>	<u>TST010</u>
Source file . . . . .	SRCFILE	>	QRPGSRC
Library . . . . .		>	TST010
Source member . . . . .	SRCMBR	>	TEST01R
Generation severity level . . .	GENLVL		<u>9</u>
Text 'description' . . . . .	TEXT		<u>*SRCMBRTXT</u>

---

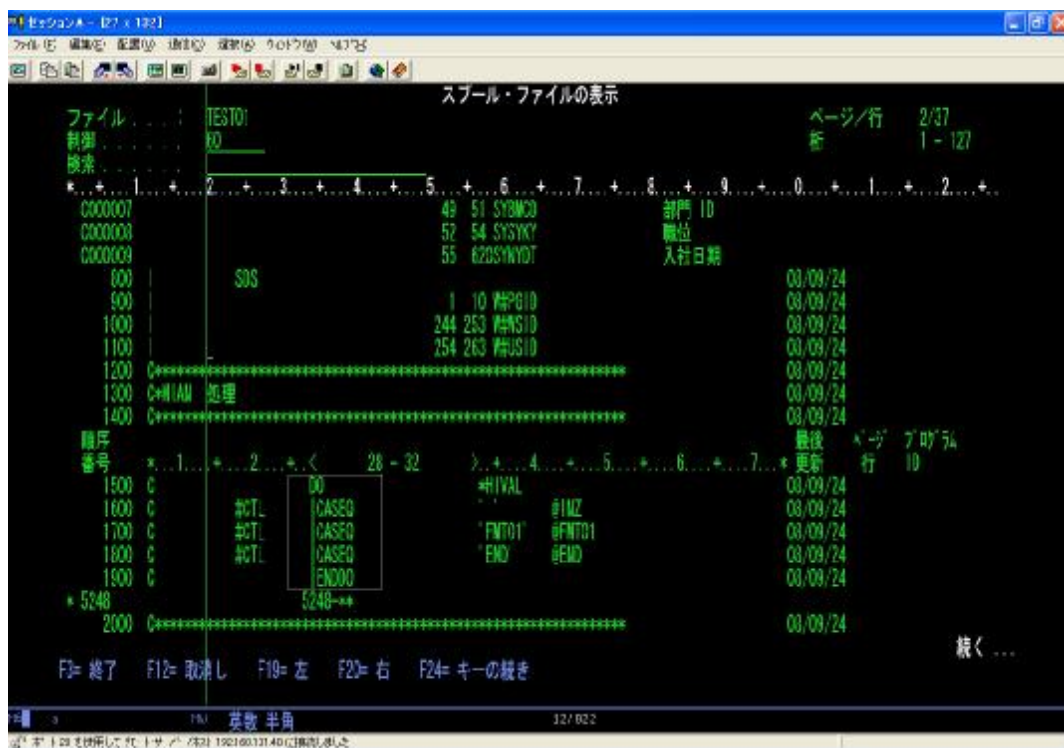
Additional Parameters

Source listing options . . . . .	OPTION		<u>          </u>
	+ for more values		<u>          </u>
Generation options . . . . .	GENOPT		<u>          </u>
	+ for more values		<u>          </u>
Source listing indentation . . .	INDENT		<u>' '</u>

More...

F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display  
F24=More keys

则编译后，在命令行键入命令 WRKJOB OPTION(\*SPLF)，找到并 5 显示编译清单，可见程序已经结构层次缩进了。如下图



### 8.1.2 获取编译清单

当编译不通过，在画面底部有提示错误信息。

使用 PDM 处理成员		S651B4CC
文件 . . . . .	<u>ORPGSRC</u>	
库 . . . . .	<u>TST010</u>	定位至 . . . . .
输入选项，按“ 执行” 键。 2= 编辑    3= 复制    4= 删除    5= 显示    6= 打印    7= 重命名 8= 显示说明    9= 保存    13= 更改文本    14= 编译    15= 建立模块 ...		
Opt	成员	类型                  文本
14	TEST01R	RPG                  雇员基本信息一览
—	TST001	RPG                  雇员基本信息一览
—	TST002	RPGLE               雇员基本信息一览
底部		
参数或命令 ===>		
F3= 退出	F4= 提示	F5= 刷新
F9= 检索	F10= 命令项	F23= 其余选项
F6= 建立		F24= 其余键
编译已停止。在文件中找到严重性级别为 30 的错误。		

在命令行键入命令 WRKJOB OPTION(\*SPLF)，找到并 5 显示编译清单



Display Spooled File									
File	TEST01R							Page/Line	1/1
Control	_____							Col umns	1 - 127
Find	_____								
*.....1.....2.....3.....4.....5.....6.....7.....8.....9.....0.....1.....2.....									
5722WDS V5R2M0 020719	IBM RPG/400				TST010/TEST01R		08/10/13 16: 29: 47	页面	
编译器	IBM RPG/400								
命令选项:									
程序	:	TST010/TEST01R							
源文件	:	TST010/QRPGSRC							
源成员	:	TEST01R							
源列表选项	:	*SOURCE	*XREF	*GEN	*NODUMP	*NOSECLVL	*NOSRCDDBG	*NOLSTDBG	
生成选项	:	*NOLI ST	*NOXREF	*NOATR	*NODUMP	*NOOPTIMIZE			
源列表缩排	:	*NONE							
类型转换选项	:	*NONE							
排序顺序	:	*HEX							
语言标识符	:	*JOB RUN							
SAA 标志	:	*NOFLAG							
生成严重性级别	:	9							
打印文件	:	*LIBL/OSYSVRT							
替换程序	:	*YES							
对象发行版	:	*CURRENT							
用户简要表	:	*USER							
权限	:	*LIBCRTAUT							
More...									
F3=Exit	F12=Cancel	F19=Left	F20=Right	F24=More keys					

编译清单分为源清单和编译时数据两部分，源清单主要有联机诊断信息，匹配字段表（如果使用了有匹配字段的 RPG 周期的话），附加的诊断信息，输出缓冲区中的字段位置，/COPY 成员表等信息。编译时数据包括：交替对照顺序记录 and 表或 NLSS 信息和表，文件传输记录，数组记录，表记录，信息总计，最终总计，代码生成报告（只在有错误时出现）等信息。

8.1.3 改正编译错误

对改正编译错误有用的编译清单主要部分有：源语句部分、附加信息部分、/COPY 表部分各类总计部分。

通过联机诊断信息和错误 ID，在源语句段上找到错误，然后改正通过 SEU 改正源语句错误，是修改程序编译错误的常用方法。

为了帮助改正编译错误，也许需要在清单中包括第二信息文本——特别对于 RPG 初学者。要想这样做的话，需在建立命令中指定 OPTION(\*SECLVL)参数。这样就会把第二级信息加到信息总计中的信息清单中。

如下图的编译清单，我们会在命令行处键入 B，然后执行，到编译清单的最后部分，查看最终总计。如下图



Display Spooled File										
File . . . . .	TEST01R						Page/Line	9/1		
Control . . . . .	B						Columns	1 - 127		
Find . . . . .										
*.....1.....2.....3.....4.....5.....6.....7.....8.....9.....0.....1.....2.....										
5722WDS V5R2M0 020719 IBM RPG/400 TST010/TEST01R 08/10/13 16:29:47 页面										
消息总结										
* QRG7030 Severity: 30 Number: 1										
消息 . . . . . : 字段或指示符未定义。										
* QRG7031 Severity: 00 Number: 13										
消息 . . . . . : 名称或指示符未被引用。										
* * * * * 消息总结结束 * * * * *										
5722WDS V5R2M0 020719 IBM RPG/400 TST010/TEST01R 08/10/13 16:29:47 页面										
最终总结										
消息计数: (根据严重性号)										
总计	00	10	20	30	40	50				
14	13	0	0	1	0	0				
程序源总计:										
记录 . . . . .	110									
规范 . . . . .	63									
表记录 . . . . .	3									
注释 . . . . .	42									
编译已停止。在文件中找到严重性级别为 30 的错误。										
* * * * * 编译结束 * * * * *										
Bottom										
F3=Exit F12=Cancel F19=Left F20=Right F24=More keys										

从错误信息统计可以看出，当前程序编译之后有 4 个 30 级错误，3 个 20 级错误，2 个 10 级错误，14 个 0 级错误。

00 级的错误仅是信息级别(Information)，可以不理睬；

10 级的错误是警告，也可以不理睬；

20、30 级别的错误就是正式的错误，也就是不解决它们，程序就无法编译通过

除此之外，还有 40 级，40 级的错误通常是程序中声明的文件不存在。

总之，级别越大，就表示错误的问题越严重，所以排错的顺序，是先大后小。即先排除 40 级错误，再来查 30 级错误，然后才是 20 级。

举个例子，如果在写代码时，将声明的文件名写错，那么 30 级错误可能会有上百个，凡是涉及到与该文件中字段有关的语句，肯定都会报错；此时排除 40 级错误（文件名声明错误）之后，会发现 30 级错误将会大减少。所以说，排错顺序是先大后小。

看到错误统计后，向上翻到联机信息。

例如，找到错误 ID QRG5177 的重大度为 30 度。下边的信息给出了错误的原因，是 D0, DOUXX, IFXX 或 SELEC 命令没有相应的结束命令匹配。在上边的检索命令处键入错误编号 5177，然后 SHIFT + F4，找到源语句段上的错误标识。通常情况下，语法错误就出现在标记的源代码行或附近，系统对于错误的提示很准，不用怀疑。（与 C 不同，C 编译后，只是说疑似某行出错）我们在根据给出的错误原因，用 SEU 进入源程序，找到对应的位置，检查命令匹配，追加匹配的结束命令即可改正此错误。

#### 8.1.4 常见错误信息

RNF2120 声明的文件不存在。

RNF7030 变量未定义，通常随着如果变量未定义，那么与该变量有关的每一句话，都会报错，并且除了 7030 之外，还会有诸如类型不匹配这些的错误，所以排错时，一般都是先修改 7030 的错误。

## 8.2 DEBUG 调试

但很多情况下可能我们来看去看，始终觉得程序并没有错误。这时，我们就可以使用 DEBUG 来进行跟踪调试。所谓跟踪调试，就是指系统运行时，我们可以逐步地运行程序，并且运行每一步时，都可以观察到程序中变量的值，这样就可以知道程序为什么运行得不理想。通常而言，一个程序的错误只要是可以重现，（也就是在相同的初始条件下，始终会出现同样的错误），那我们就一定可以将错误 DEBUG 找出来；这时最重要的，就是找出程序出错的规律来。就算没错误的时候，对正常运行的程序进行 DEBUG，也可以更好地帮助理解程序的运行方式。

总而言之，DEBUG 是个很有用的调试方法，做为一个 RPG 程序员，不能不掌握它。

### 8.2.1 调试时的入口参数问题

在测试阶段，在命令行处无论调用执行（CALL）程序，还是调试（DEBUG）程序，当参数中有数值型参数时，都存在入口参数的传入问题。如果写一个调用程序（如测试 CL），在程序中，先给对应类型变量赋值，利用变量的形式传递，可以解决，否则直接像字符型一样传入，会导致传入不成功或参数错误。下面通过几个实例，讲解一下直接传递数值型参数的方法（字符型直接赋值即可）。

设程序 PGMA 有参数 Parm1, Parm2:

1) parm1(7S 0) = 276, parm2(7S 2) = 15.73

传递方式: CALL PGMA PARM('0000276' '0001573')

2) parm1(7S 0) = -276, parm2(6S 2) = -15.73

传递方式: CALL PGMA PARM('-000276' '-01573')

3) parm1(7P 0) = 276, parm2(7P 2) = 15.73

传递方式: CALL PGMA PARM( X'0000276F' X'0001573F')

4) parm1(6P 0) = 276, parm2(6P 2) = 15.73

传递方式: CALL PGMA PARM( X'F000276F' X'F001573F')

5) parm1(7P 0) = -276, parm2(6P 2) = -15.73

传递方式: CALL PGMA PARM( X'0000276B' X'F001573B')

由此可见：对于 S 类型的参数，只要按照字符串的形式，在没有数字的位置补 0，不用输入小数点，就可以得到正确的结果，如果是负数，在第一位输入负号 - 即可，但是会占用一个数字位，7 位的负数就只能输入 6 位数字了。对于 P 类型的参数，数字部分按照 S 类型的一样输入，还需要在字符串前加 X，并且如果长度为奇数，需要给字符串加 F 后缀，如果长度为偶数，需要在字符串的前后都加 F。如果是负数的，需要把字符串后面的 F 变为 B。（X 表示按 16 进制取值，由于 P 类型的数据存储方式是半个字节存储一个数字，所以需要在前后加其它的字母补齐空位。）

### 8.2.2 RPG 的调试

在命令行键入 STRISDB 程序名，然后按 F4，UPDPROD 项选为“ \*YES” ，程序入口参数根据程序实际需求设定，执行。



如果我们要调试的程序（B）是在某些程序（A）中被调用的程序，而我们只想调试这个被调用的程序(B)时，我们也可以把 INVPGM 项选为“ \*NO” ,然后执行，出现命令入力画面，键入执行程序 A 的命令。执行即可。

Start ISDB (STRISDB)

Type choices, press Enter.

Program . . . . .	> <u>TEST01R</u>	Name
Library . . . . .	> <u>TST010</u>	Name, *CURLIB, *LIBL
Update production files . . . . .	> <u>*YES</u>	*YES, *NO
Invoke program . . . . .	> <u>*YES</u>	*YES, *NO, *CMD
Parameters for call . . . . .	<u>                    </u>	

+ for more values

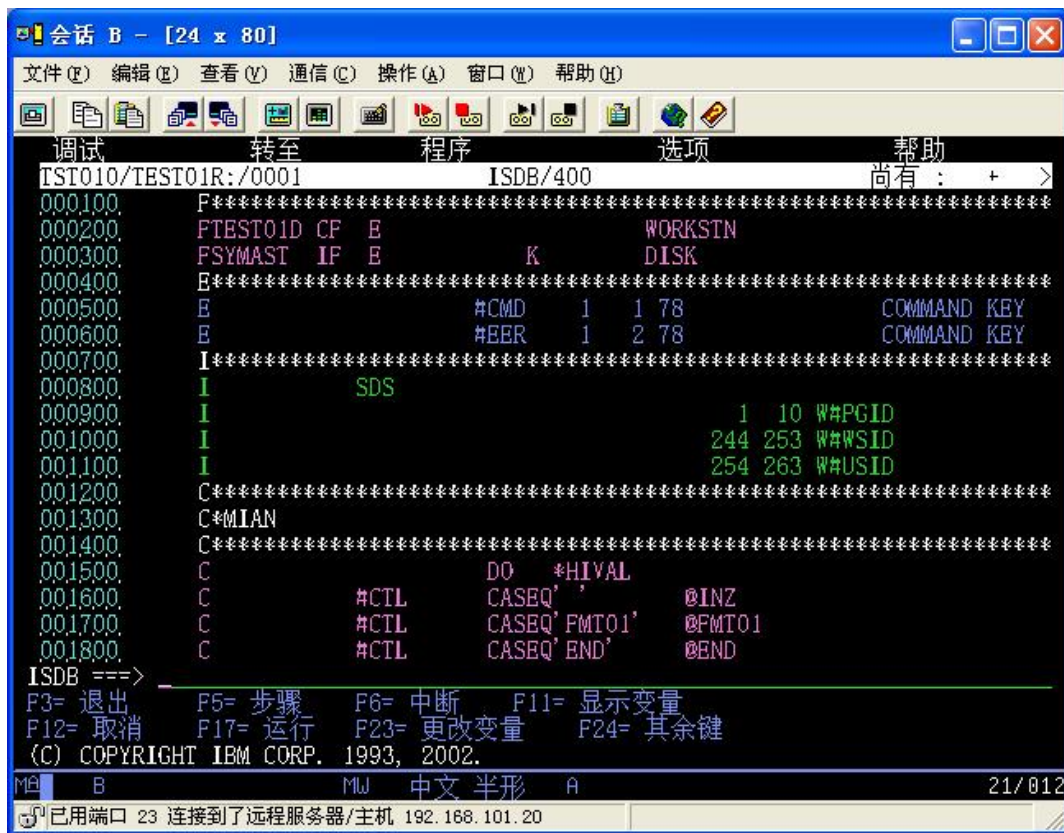
Additional Parameters

Source member . . . . .	<u>*PGM</u>	Name, *PGM
Source file . . . . .	<u>                    </u>	Name
Library . . . . .	<u>                    </u>	Name, *CURLIB, *LIBL
Job to service . . . . .	<u>*                    </u>	*, *SELECT

Bottom

F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display  
F24=More keys

接下来，就可以进入 DEBUG 模式。也就是我们可以看到这个程序编译后的可执行代码。比起我们写的源代码而言，根据式样的不同，颜色上也做出了区别，如下图：



The screenshot shows the hiSoft debugger window titled "会话 B - [24 x 80]". The menu bar includes 文件(F), 编辑(E), 查看(V), 通信(C), 操作(A), 窗口(W), and 帮助(H). The toolbar contains icons for file operations, editing, and debugging. The main window displays assembly code for the program "TEST010/TEST01R:/0001" in "ISDB/400" mode. The code is shown in a list format with addresses from 000100 to 001800. The code includes instructions like FTEST01D, PSYMAST, E, I, C, and various control flow instructions. A command menu is visible at the bottom of the code window, listing functions like F3=退出, F5=步骤, F6=中断, F11=显示变量, F12=取消, F17=运行, F23=更改变量, and F24=其余键. The status bar at the bottom shows "已用端口 23 连接到了远程服务器/主机 192.168.101.20" and "21/012".

接下来就可以设置断点。所谓断点，也就是当我们在 DEBUG 环境下执行 CALL 程序时，系统执行到断点处将会停止运行的代码行，然后等待我们输入执行命令（如 F5 是单步向下执行、F3 是直接退出运行等）。如果没有设置断点，在 DEBUG 环境下执行 CALL 命令，系统将会直接将程序运行完毕。

DEBUG 模式中设置断点的方法，是在对象代码行按 F6 键。当代码行被设置断点后，将会反白显示。（注意，只有有效可执行代码可以设置断点）如果觉得断点设置得不对，想取消，那么在该行再按一下 F6 键，代码行将会取消反白显示，也即是取消该行断点。

在 DEBUG 模式中，可做如下操作：

- F3 退出程序（不执行下面的语句了）
- F4 显示到当前执行代码画面。
- F5 单步向下执行程序
- F6 定位断点（定位断点处，会反白显示）/ 已定位断点处再按 F6，即是取消断点
- F9 在下边的命令行键入的复显。
- F11 查看当前光标处的变量的值
- F12 回到前画面
- F13 执行到当前光标所在行，并从此行开始执行。
- F14 新增调试调用程序。
- F17 程序运行到下一断点处（若无断点，则程序会运行完毕）

F23 调出交互式命令行，可以变更光标位置变量。

在 DEBUG 模式的命令行中执行：

F 字符串 顺序向下查找字符串（如 F D9CMD，光标将会跳至顺序向下，代码段中首个 D9CMD 字符串处

F16 续查找下一个字符串（紧接在 F 字符串后使用）

### 8.2.3 RPGLE 的调试

如果想对 RPGLE 程序进行调试，必须在编译时指定参数：

编译程序时，先按 F4，再按 F10，修改编译参数的值(可能需要翻页)：DBGVIEW 项填“ \*LIST”，或“ SOURCE”。如果编译的程序未指定这个参数，那么执行 STRDBG 命令时，将会毫无效果。

Create Bound RPG Program (CRTBNDRPG)			
Type choices, press Enter.			
Debugging views . . . . .	<u>*LIST</u>	*STMT, *SOURCE, *LIST...	
Output . . . . .	<u>*PRINT</u>	*PRINT, *NONE	
Optimization level . . . . .	<u>*NONE</u>	*NONE, *BASIC, *FULL	
Source listing indentation . . .	<u>*NONE</u>	Character value, *NONE	
Type conversion options . . . .	<u>*NONE</u>	*NONE, *DATETIME, *GRAPHIC...	
+ for more values			
Sort sequence . . . . .	<u>*HEX</u>	Name, *HEX, *JOB, *JOB RUN...	
Library . . . . .		Name, *LIBL, *CURLIB	
Language identifier . . . . .	<u>*JOB RUN</u>	*JOB, *JOB RUN...	
Replace program . . . . . >	<u>*NO</u>	*YES, *NO	
User profile . . . . .	<u>*USER</u>	*USER, *OWNER	
Authority . . . . .	<u>*LIBCRTAUT</u>	Name, *LIBCRTAUT, *ALL...	
Truncate numeric . . . . .	<u>*YES</u>	*YES, *NO	
Fix numeric . . . . .	<u>*NONE</u>	*NONE, *ZONED, *INPUTPACKED	
Target release . . . . .	<u>*CURRENT</u>	*CURRENT, *PRV, V4R5M0, ... More...	
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display F24=More keys			

在命令行键入 STRDB 程序名，然后按 F4，UPDPROD 项选为“ \*YES”，执行，接下来，就可以进入源代码段，我们要做的是设置断点。设置断点的方法与 RPG 的操作相同。设置好断点后，F12 退出设置断点，在命令行出，键入调用（CALL）要调试的程序命令。就可以进入 DEBUG 模式了。

在 DEBUG 模式中，可做如下操作：

F3 退出程序（不执行下面的语句了）

F6 定位断点（定位断点处，会反白显示）/ 已定位断点处再按 F6，即是取消断点

F9 在下边的命令行键入的复显。

F10 单步向下执行程序

F11 查看当前光标处的变量的值

F12 程序运行到下一断点处（若无断点，则程序会运行完毕）

F13 断点停止的条件处理。

F14 新增调试调用程序。

F17 对光标所在变量进行监视。如果发生变化，会提示。

F18 对监视的变量进行处理。

还有一些操作，可以参照 DEBUG 模式下边提示帮助信息进一步了解。

在 DEBUG 模式的命令行中执行：

F 字符串：顺序向下查找字符串（如 F D9CMD，光标将会跳至顺序向下，代码段中首个 D9CMD 字符串处 F16 续查找下一个字符串（紧接在 F 字符串后使用）

EVAL 变量名：查看当前变量的值；（如 EVAL FLD01，就是查看字段 FLD01 的值）

EVAL 变量名=' '，直接更改当前变量的值（通常不要使用这个功能）

W 变量名：对变量进行监视。

当我们结束 DEBUG 调试之后，必须输入“ ENDDBG” 用来结束 DEBUG 调试模式。否则无法进行下次 DEBUG 操作，而且可能会带来一些不可预知的错误。

#### 8.2.4 批处理程序的调试

通常我们对于批处理程序，在交互式状态下测试好以后，再正式开始使用提交后台的方式，让程序进行批处理运行，但有时交互式环境与批处理环境的不同，会导致程序运行异常，此时使用以下方法跟踪批处理程序会更为直接。

1. 以 HOLD(\*YES)参数提交 JOB 到 QBATCH JOB 中，让 JOB 暂时挂起；  
如：SBMJOB CMD(CALL PGM(TEST04R)) JOB(TEST04R) JOBQ(QBATCH) HOLD(\*YES)
2. 使用 WRKSBMJOB 查看所提交的 JOB 的以下 3 个参数值：  
\*Job id  
\*User Name  
\*Job Number

Work with Submitted Jobs						S651B4CC
						08/10/13 16:57:07
Submitted from . . . . . : *USER						
Type options, press Enter.						
2=Change 3=Hold 4=End 5=Work with 6=Release 7=Display message						
8=Work with spooled files						
Opt	Job	User	Type	-----Status-----	Function	
<u>5</u>	TEST04R	TST010	BATCH	JOBQ	HELD	
						Bottom
Parameters or command						
====> _____						
F3=Exit	F4=Prompt	F5=Refresh	F9=Retrieve	F11=Display schedule data		
F12=Cancel	F17=Top	F18=Bottom				

Work with Job				System:	S651B4CC
Job:	TEST04R	User:	TST010	Number:	023657
Select one of the following:					
1. Display job status attributes					
2. Display job definition attributes					
3. Display job run attributes, if active					
4. Work with spooled files					
10. Display job log, if active or on job queue					
11. Display call stack, if active					
12. Work with locks, if active					
13. Display library list, if active					
14. Display open files, if active					
15. Display file overrides, if active					
16. Display commitment control status, if active					
					More...
Selection or command					
====> _____					
F3=Exit	F4=Prompt	F9=Retrieve	F12=Cancel		

如图，刚刚提交的这个批处理\*Job id: TEST04R, \*User Name: TST010, \*Job Number: 023657

3. 执行 STRSRVJOB, 填入第 2 步骤获得的 3 个参数进行 QBATCH JOB



Start Service Job (STRSRVJOB)		
Type choices, press Enter.		
Job name . . . . .	<u>TEST04R</u>	Name
User . . . . .	<u>TST010</u>	Name
Number . . . . .	<u>023657</u>	000000-999999
Bottom		
F3=Exit	F4=Prompt	F5=Refresh
F10=Additional parameters	F12=Cancel	
F13=How to use this display	F24=More keys	
Parameter JOB required.		

STRSRVJOB JOB(022748/D07103/TEST04)

#### 4. 执行 STRDBG 开始 DEBUG;

Start Debug (STRDBG)		
Type choices, press Enter.		
Program . . . . .	<u>TEST04R</u>	Name, *NONE
Library . . . . .	<u>TST010</u>	Name, *LIBL, *CURLIB
+ for more values		
Default program . . . . .	<u>*PGM</u>	Name, *PGM, *NONE
Maximum trace statements . . . .	<u>200</u>	Number
Trace full . . . . .	<u>*STOPTRC</u>	*STOPTRC, *WRAP
Update production files . . . . .	<u>*YES</u>	*NO, *YES
OPM source level debug . . . . .	<u>*NO</u>	*NO, *YES
Service program . . . . .	<u>*NONE</u>	Name, *NONE
Library . . . . .		Name, *LIBL, *CURLIB
+ for more values		
More...		
F3=Exit	F4=Prompt	F5=Refresh
F10=Additional parameters	F12=Cancel	
F13=How to use this display	F24=More keys	

执行后显示源代码;



Display Module Source										
Program:	TEST04R	Library:	TST010	Module:	TEST04R					
1	*MODULE ENTRY AND MAIN PROCEDURE ENTRY									
2	1 F*****								081008	
3	2 FSYMAST UF E K DISK								081008	
4	3 C*****								081008	
5	4=ISYMASTR								081008	
6	5=I	A	1	10	SYSYCD	雇员 I D	081008			
7	6=I	A	11	30	SYSYNM	雇员名	081008			
8	7=I	A	31	34	SYSYBT	性别	081008			
9	8=I	S	35	42	OSYSNDT	出生年月日	081008			
10	9=I	S	43	45	OSYSNTY	身高	081008			
11	10=I	S	46	48	OSYTAYX	体重	081008			
12	11=I	A	49	51	SYBMCD	部门 ID	081008			
13	12=I	A	52	54	SYSYKY	职位	081008			
14	13=I	S	55	62	OSYNYDT	入社日期	081008			
15	14 C	EXSR	@MAIN						081008	
16	15 C	EXSR	@END						081008	
17	16 C*****								081008	
18	17 C	@MAIN	BEGSR						081008	
Debug . . .									More...	
F3=End program F6=Add/Clear breakpoint F9=Retrieve F10=Step F11=Display variable F12=Resume F14=Work with module list F15=Select view F16=Repeat find F17=Watch variable F18=Work with watch F21=Command entry F22=Step into F24=More keys										

## 5. 利用 F21（SHIFT + F9）键切换到命令行；

Display Module Source										
Program:	TEST04R	Library:	TST010	Module:	TEST04R					
1	*MODULE ENTRY AND MAIN PROCEDURE ENTRY									
2	1 F*****								081008	
3	2 FSYMAST	UF	E	K	DISK				081008	
4	3 C*****								081008	
5	4=ISYMASTR									
6	5=I	A	1	10	SYSYCD	雇员 I D				
7	6=I	A	11	30	SYSYNM	雇员名				
8	7=I	A	31	34	SYSYBT	性别				
9	8=I	S	35	42	OSYSNDT	出生年月日				
10	9=I	S	43	45	OSYSNTY	身高				
11	10=I	S	46	48	OSYTAYX	体重				
12	11=I	A	49	51	SYBMCD	部门 ID				
13	12=I	A	52	54	SYSYKY	职位				
14	13=I	S	55	62	OSYNYDT	入社日期				
15	14 C	EXSR	@MAIN						081008	
16	15 C	EXSR	@END						081008	
17	16 C*****									081008
.....										
:	Command								:	
:									:	
:	==>								:	
:	F4=Prompt	F9=Retrieve	F12=Cancel							:
:									:	

在命令行执行 WRKSBMJOB，使用 6=Release 释放挂起的第 1 步骤提交的 JOB，然后系统允许你按 F10 输入 DEBUG 命令（注意：不要键入执行，否则在设立断点之前键入执行，程序就会运行，因而无法进行 debug 断点设置）；



Start Serviced Job			
Job:	TEST04R	User:	TST010
Number:	023657	System:	XXXXXXXX
The serviced job has been released from the job queue. Press Enter to start the job or F10 to enter debug commands for that job.			
Press Enter to continue.			
F10=Command entry			
(C) COPYRIGHT IBM CORP. 1980, 2002.			

6. 在 OS/400 命令行窗口；执行 DSPMODSRC 后，可通过 F6 设置断点；然后按 F3 退出，再按 F12 退出命令行；

Command Entry		XXXXXXXX
Previous commands and messages:		Request level: 8
(No previous commands or messages)		
Type command, press Enter.		Bottom
===> <u>DSPMODSRC</u>		
F3=Exit F4=Prompt F9=Retrieve F10=Include detailed messages		
F11=Display full F12=Cancel F13=Information Assistant F24=More keys		

Display Module Source																
Program:	TEST04R	Library:	TST010	Module:	TEST04R											
1	*MODULE ENTRY AND MAIN PROCEDURE ENTRY															
2	1 F*****								081008							
3	2 FSYMAST UF E K DISK								081008							
4	3 C*****								081008							
5	4=ISYMASTR															
6	5=I	A	1	10	SYSYCD	雇员 I D										
7	6=I	A	11	30	SYSYND	雇员名										
8	7=I	A	31	34	SYSYBT	性别										
9	8=I	S	35	42	OSYNDT	出生年月日										
10	9=I	S	43	45	OSYSNTY	身高										
11	10=I	S	46	48	OSYTAYX	体重										
12	11=I	A	49	51	SYBMCD	部门 ID										
13	12=I	A	52	54	SYSYKY	职位										
14	13=I	S	55	62	OSYNYDT	入社日期										
15	14 C	EXSR	@MAIN						081008							
16	15 C	EXSR	@END						081008							
17	16 C*****								081008							
18	17 C @MAIN BEGSR															081008
Debug . . .																
F3=End program F6=Add/Clear breakpoint F9=Retrieve F10=Step F11=Display variable F12=Resume F14=Work with module list F15=Select view F16=Repeat find F17=Watch variable F18=Work with watch F21=Command entry F22=Step into F24=More keys Breakpoint added to line 15.																

7. 键入执行释放挂起的 JOB；程序将在断点中停留；可以使用交互式 DEBUG 使用 DEBUG 命令进行处理；

Display Module Source										
Program:	TEST04R	Library:	TST010	Module:	TEST04R					
1	*MODULE ENTRY AND MAIN PROCEDURE ENTRY									
2	1 F*****								081008	
3	2 FSYMAST UF E K DISK								081008	
4	3 C*****								081008	
5	4=ISYMASTR									
6	5=I		A	1	10	SYSYCD	雇员 I D			
7	6=I		A	11	30	SYSYND	雇员名			
8	7=I		A	31	34	SYSYBT	性别			
9	8=I		S	35	42	OSYSNDT	出生年月日			
10	9=I		S	43	45	OSYSNTY	身高			
11	10=I		S	46	48	OSYTAYX	体重			
12	11=I		A	49	51	SYBMCD	部门 ID			
13	12=I		A	52	54	SYSYKY	职位			
14	13=I		S	55	62	OSYNYDT	入社日期			
15	14 C		EXSR	@MAIN					081008	
16	15 C		EXSR	@END					081008	
17	16 C*****								081008	
18	17 C	@MAIN	BEGSR							081008
										More...
Debug . . .										
F3=End program F6=Add/ClearBreakpoint F9=Retrieve F10=Step F11=DisplayVariable F12=Resume F14=Work with module list F15=Select view F16=Repeat find F17=Watch variable F18=Work with watch F21=Command entry F22=Step into F24=More keys Breakpoint added to line 15.										

8. 程序或者 JOB 结束后，一定要使用 ENDDBG 和 ENDSRVJOB 结束操作。

注意：

如果在批处理作业的 RPGIV 程序中出现交互语句，如：DSPLY；显示文件输入输出语句程序的调用，如：EXFMT，作业将会被挂起处于 MESSAGE WAIT 状态，这是因为批处理作业无法处理显示信息而引起的。

## 第九章 数据库操作

### 9.1 Query

Query/400 是 IBM 的一个特许程序，此程序支持从 AS/400 数据库中获得信息来设计应用。它可以从任一由 OS/400 DDS, OS/400 IDDU, SQL/400 定义的数据库文件中获取信息。

使用 Query 从一个或更多个数据库文件中去选择、重排和分析信息(数据)来生成报表和其它数据文件。能生成自己的查询定义并运行它们，可以运行已有的，不是你生成的查询，或者可以用对某个数据库文件运行一个缺省的查询(使用非命名查询)。你来确定查询将检索什么样的数据，报表的格式，以及它是否显示、打印、或送到其它数据库文件中。

能使用 Query 从一个文件或一组（至多 32 个）文件获取信息，能选取所有的字段、一部分字段把它们按你要求的输出类型组织起来。可以在输出中包含文件的所有的记录，也可用记录选择测试，仅包含一部分记录。

本节开始介绍查询的基本信息，接着将介绍使用 Query 能完成的主要任务(例如生成、显示、或运行查询)。

#### 9.1.1 查询的基本概念

系统中的一些部件能组织和存贮信息或数据，这样，你和其它系统用户能用它得到想要的结果。下面将介绍这些部件，说明它们的情况以及它们与 Query 的关系，并指导你在哪儿能找到更多的有关内容。

文件、字段和记录格式：信息或数据用多种样式组织和存贮在系统中，其主要对象叫做数据库文件(经常被称为文件)。一个文件中包含的相对独立的信息单元叫做记录，每个都包含相关的数据段，记录中的每个信息段叫做一个字段，用记录格式定义字段的组织(经常叫做格式)。

在运行查询来产生报表时，Query 使用文件、字段和记录格式来得到想从数据库记录格式中得到的信息，并用这些记录产生查询报表。

#### 9.1.2. 使用查询命令

一条命令是请求一个系统功能的语句，这意味着你只需要记住几个字符的命令而不用记住所有的各个指令或花费时间走过一组菜单。

查询有四条能从命令行键入的命令：

STRQRY 产生查询菜单。

WRKQRY 产生‘处理查询’显示。

RUNQRY 运行已存在的查询，或运行一个缺省查询，这个命令可以嵌套在 CL 程序中运行几个查询。

DLTQRY 删除一个或多个查询定义。

#### 使用查询菜单

查询菜单允许选择查询任务来处理查询、运行以前定义的查询、删除查询定义或处理文件。

## STRQRY 进入 QUERY 的主菜单

QUERY	Query Utilities	System: XXXXXXXX
Select one of the following:		
Query for AS/400		
1. Work with queries		
2. Run an existing query		
3. Delete a query		
DB2 for AS/400		
10. Start DB2 Query Manager for AS/400		
Query management		
20. Work with query management forms		
21. Work with query management queries		
22. Start a query		
23. Analyze a Query for AS/400 definition		
		More...
Selection or command		
===> _____		
F3=Exit F4=Prompt F9=Retrieve F12=Cancel F13=Information Assistant		
F16=AS/400 Main menu		

选择第 1 项（处理查询），并按 Enter 键，出现‘处理查询’的显示。在此显示中可以指定使用的查询及所用方法。能用该显示开始的任务（包括运行和删除查询的其他方法）。

### 使用查询

输入选项，按“ 执行” 键。

选项 . . . . .

1= 创建， 2= 更改， 3= 复制， 4= 删除

5= 显示， 6= 打印定义

8= 以批处理方式运行， 9= 运行

查询 . . . . .

名称， F4 显示列表

库 . . . . .

XXXXXXXXXX

名称， \*LIBL, F4 显示列表

F3= 退出

F4= 提示

F5= 刷新

F12= 取消

(C) COPYRIGHT IBM CORP. 1988

#### 查询任务 任务说明

1=生成 生成（定义）一个新查询。

2=修改 修改已存在的查询定义。

3=拷贝 拷贝一个已存在的查询定义。

4=删除 删除一个已存在的查询定义。

5=显示 显示查询定义但不能修改，（要修改查询，选 2）。

6=打印定义 打印一个查询定义。

8=用批处理运行 用批处理运行一个查询，减少由交互运行查询出现的资源消耗。

9=运行 运行一个查询。根据查询定义中的规定内容，可以显示一个报表，打印一个报表，或把数据送入数据库文件中。

选择了第 2 项（运行一个已有的查询），则出现 RUNQRY 命令的提示显示。可用这条命令运行查询或从一个或多个数据库文件中选定的数据生成一个查询报表。可用该命令运行一个已存在的查询（一个已被定义且用名字存贮在系统中的查询），或运行一个“ 缺省” 查询（一个未命名且大部分使用由系统提供值的查询）。假如不知道查询名或库名，使用选项 1（处理查询），则可以得到查询和库的列表。

Run Query (RUNQRY)		
Type choices, press Enter.		
Query . . . . .	<u>          </u>	Name, *NONE
Library . . . . .	<u>*LIBL</u>	Name, *LIBL, *CURLIB
Query file:		
File . . . . .	<u>          </u>	Name, *SAME
Library . . . . .	<u>*LIBL</u>	Name, *RUNOPT, *LIBL, *CURLIB
Member . . . . .	<u>*FIRST</u>	Name, *RUNOPT, *FIRST, *LAST
+ for more values		
Report output type . . . . .	<u>*RUNOPT</u>	*RUNOPT, *DISPLAY...
Output form . . . . .	<u>*RUNOPT</u>	*RUNOPT, *DETAIL, *SUMMARY
Record selection . . . . .	<u>*NO</u>	*NO, *YES
Bottom		
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display		
F24=More keys		

假如选择第 3 项（删除查询），出现 DLTQRY 命令的提示显示。可以用此命令删除一个查询定义。假如不知道一个查询名或库名，使用选项 1（处理查询），可以得到一张查询和库的列表。

Delete Query (DLTQRY)		
Type choices, press Enter.		
Query . . . . .	<u>          </u>	Name
Library . . . . .	<u>*LIBL</u>	Name, *LIBL, *CURLIB...
Bottom		
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display		
F24=More keys		

定义查询			
查询 . . . . . :		选项 . . . . . :	CREATE
库 . . . . . :	QGPL	CCSID . . . . . :	937
输入选项, 按“ 执行” 键。按 F21 可全部选中。			
1= 选择			
Opt	查询定义选项		
1	指定文件选择		
-	定义结果字段		
-	选择字段并对字段排序		
-	选择记录		
-	选择排序字段		
-	选择整理顺序		
-	指定报告列格式		
-	选择报告摘要函数		
-	定义报告中断		
-	选择输出类型和输出形式		
-	指定处理选项		
F3= 退出      F5= 报告      F12= 取消			
F13= 布局      F18= 文件      F21= 全部选中			

为了从查询定义选项列选取使用的选项, 在这些选项的 OPT 列键入 1, 然后按 Enter 键, 所选择的选项的显示一个接一个的给出, 这样就可以生成定义的各部分。

以下简要介绍每个定义步骤:

#### 规定文件选项

这个选项是必须的, 用它去规定查询获得信息所用的一个或多个文件。如果指定多个文件, 会有显示给出让你指定文件如何连接。

#### 定义结果字段

使用这个选项去定义字段, 它不在文件中, 但查询要使用的字段。例如, 选择的文件包括表示周数的字段, 但它不包括表示天数的字段。而想在报表中显示天数, 而不是周数, 那么可以定义一个结果字段, 用它来放用周数来确定天数计算的结果。

#### 选择和排序字段

用它来选择字段(从选择的文件和结果字段中), 让它出现在报表中, 也可以规定它们出现的顺序。

#### 选择记录

用这个选项从一个或多个文件中选择记录, 这样报表中仅包括选择的记录。

#### 选择分类字段

用这项去规定字段如何分类, 这样记录用特殊顺序出现(例如用字母顺序, 用升序或降序)。



### 选择整理顺序

使用这个选项去选择查询的整理顺序。选择的整理顺序，会影响查询中许多不同的事情，包括记录选择和记录在分类时的顺序。这个整理顺序通常与你的母语有关，能对每个查询指定不同的语言。当首次定义查询时能设置缺省的整理顺序，这样就不必再修改。

### 指定报表列的格式

用此选项去修改报表中列标题、列、间隔、数字编辑、长度和小数位。

### 选择报表汇总功能

使用此选项规定报表中为每个字段规定一个或多个汇总功能类型：总计、平均值、最小值、最大值和计数。

### 定义报表分段

使用此项去规定怎样把报表在记录中分组。

### 选择输出类型和格式

用这个选项规定是否输出要显示、打印或送给数据库文件。如果要打印，要设置打印属性，也可以用它去规定是否要明细或总计输出。

### 设定处理选项

用此选项规定查询计算结果是否可截断或者舍入，如果想忽略十进数据错误或者在转换过程中忽略字符替换错误，可以使用此选项。

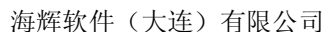
可以指定一个、多个或全部显示的选项，要选择所有的选项，用 F21 键。

## 9.2 SQL 在 AS400 上的应用实例

DB2 UDB for iSeries 与其他数据库一样，支持 SQL 语言。同时也提供了使用 SQL 集成工具。在这里，我们通过实例，简述 SQL 工具的启动和使用方法。

### 9.2.1 SQL 工具的启动

在命令行，键入 STRSQL，即可启动 SQL 工具，如下图：



在上图，我们输入 SQL 语句，执行即可。也可以按照按 F4 提示，查看 SQL 命令交换画面。SQL 工具还提供了很多方便使用的功能，可以参照界面下方的帮助提示，这里就不赘述。

在 SQL 输入行，键入如下 SQL 语句，执行

即可查询 TST010 库下 SYMAST 表中的所有数据。数据的添加，修改，删除操作与此类似，这里就不在赘述。

### 9.3 事务处理 COMMIT

COMMIT 提交当前事务。所有事务的更改都将为其他事务可见，而且保证当崩溃发生时的可持续性。在程序中经常使用 COMMIT 来控制事物处理的完整性，保证在发生异常的情况下数据的完整性和正确性。

访问数据库文件时使用 COMMIT/ROLLBK，可以确保对数据库文件操作的一致性：全部数据库文件操作都是成功的 COMMIT 提交；全部数据库文件操作都是无效的 ROLLBK 回滚。用这种方法，从而保证了数据库文件的完整性；另外，可以把一组操作当作一个单元来处理。

举个实际的例子，如果程序中，以修改的方式声明了四个文件 A、B、C、D，其中 A、B、C 都使用了 COMMIT 关键字，而 D 未使用 COMMIT 关键字；在程序执行过程中，首先更改了 A、B、D 的值

在接下的处理中，如果业务流程判断，逻辑有误，不再执行，此时可以进行回滚操作，此时 A、B 的数据恢复成为修改之前的数值；D 的数据，因为在声明文件时未使用 COMMIT 关键字，所以回滚操作对它无效，即 D 的数据仍然保持修改之后的值；

而如果业务流程判断正常，程序顺利执行完毕，此时需要进行一次确认操作（COMMIT），来落实数据的修改。

通常，我们称 COMMIT 为提交确认操作（也可称之为落实操作），与回滚操作，统称为“事务处理”

#### 使用方法

要实现事务处理的功能，需要以下的准备工作：

- 1、生成一个日志接收器（CRTJRNRCV）
- 2、生成一个日志，并将该日志连接之前建立的日志接收器（CRTJRN）
- 3、将需要声明的文件，增加到上面已建立的日志中（STRJRNPF），只增加 PF 文件即可，不需要增加相应的 LF 文件；一个 PF 文件只用增加一次。
- 4、对于当前进程，执行启用日志的命令 STRCMTCTL LCKLVL（\*ALL）（对于同一进程，执行一次即可，通常在签到时由公共程序来执行）
- 5、在程序中，对声明的文件，添加 COMMIT 关键字
- 6、在执行完修改操作语句后，执行事务处理。

步骤 1、2、3 通常由系统管理人员来执行，对于普通程序员而言，不需要理会。不过需要注意，一个 PF 文件如果重新编译之后，系统会自动将它从日志中去掉，所以在重新编译过 PF 文件之后，需要执行 STRJRNPF 命令将它再加入到日志中去。（当然，新增的 PF 文件，如果需要进日志处理，也是要加到日志中去的）

而步骤 4，一般是隐含在登录程序中执行的。登录程序可以 CHGUSRPRF 命令来查看。再引申一下，对于使用 SBMJOB 命令提交后台执行的程序而言，因为 SBMJOB 是提交产生了一个新进程，这个新进程未执行签到程序，所以如果运行的程序有用到事务处理的话，将会报错。解决之道是在这个新进程中，执行业务处理程序之前，先执行一次签到程序，或执行 STRCMTCTL 命令。

#### 注意事项

- 1、在实际使用过程中，普通的业务处理程序，通常会将所有的声明为“修改”方式打开的文件，都加上 COMMIT 关键字，所谓一损俱损，一荣俱荣，只要业务逻辑判断上有任何一处不

符合要求，所有已修改的文件，都恢复成为修改之前的值，这样的处理是完全正确，也是应该的；但对应于批处理程序，尤其是循环处理某个表，或某几个表中的多笔记录时，通常会采取处理完一笔记录，执行一次事务处理的方式，因为我们不希望因为一笔记录不符合要求，就导致其它所有记录都不再处理。当然，有特殊要求的除外。

- 2、为了便于排错，跟踪数据，通常日志性的文件（仅执行写操作），是不会定义 COMMIT 关键字的。
- 3、落实操作与回滚操作，有两种方式来实现，一种方式是在程序中使用操作码，对应操作码为 COMMIT 与 ROLBK；第二种方式是使用命令，对应的命令为 COMMIT 与 ROLLBACK。这两种方式效果是相同的。
- 4、事务处理仅针对当前进程的，对其它进程无效。
- 5、声明的文件如果定义了 COMMIT 关键字，那么对其进行修改后，修改的记录在执行事务操作之前，会一直保持锁定状态。锁定的记录越多，其它程序的执行效率将会越低，同时当前进程在执行回滚操作时的时间就越长，所以原则上，我们通常在完成了一次处理之后，尽快执行事务操作。
- 6、在交互式画面中，我们签退时（SIGNOFF），系统会默认执行一次 ROLLBACK 操作（这个好象可选，可以改为默认执行 COMMIT 操作，待查）。所以有时如果发现数据莫名其妙的恢复了，或非所预期的结果，不要首先怀疑系统故障或公共程序，请先回想一下自己的操作是否合理。就概率而言，在平常开发、测试时，建议多用 ROLLBACK 命令，但运行时间超长的大程序不适用此原则。

SQLRPGLE 程序中，对于 SQL 语句修改的文件，程序默认对文件进行了 COMMIT 声明，也就是执行了 SQL 语句之后，需要进行事务处理操作。否则执行完程序后，如果仅检查数据更新无误，在未执行事务处理操作的情况下就签退的话，系统将会默认执行 ROLLBACK 操作，导致修改无效。

#### 9.4 关于锁表的问题 LOCK

通常的锁表，是指锁记录。

##### 程序内部调用之一

如果 A 程序调用 B 程序，而且 A、B 程序都用 U 的方式打开同一个文件，更改同一条记录时，那么在调用 B 程序之前，需要有一个 UPDATE 或者 CLOSE 的动作，否则在被调用的 B 程序将会锁表。

##### 程序内部调用之二

单个程序内，无论有无定义 COMMIT，只要做了 UPDATE 的操作，都可以多次对同一条记录进行定位操作（CHAIN、READ 等）的；只要没做 UPDATE 操作，下一次做定位操作时，就会锁表。

##### 程序之间的调用

A 程序定义了 COMMIT，在定位操作语句（CHAIN、READ 等）之后，UPDATE 操作之前，其它程序查找该条记录，仍然是原始的数据；在 UPDATE 操作之后，事件处理语句之前（COMMIT、ROLBK），其它程序查找该条记录，则是更改后的数据了。如果之后再继续进行 ROLLBACK 操作，其它程序查到的就又是原始的数据。这样在统计时就会出现错误，所以说统计程序一定要在事务处理语句之后进行。

当 A 程序在执行 UPDATE 操作之后,事务处理语句之前,其它程序以修改的方式读取该条记录时,是会锁表的。也就是当声明的文件定义了 COMMIT 关键字时,在打开一条记录之后,除了进行 UPDATE 或 UNLOCK 语句解锁之外,还必须进行事务处理操作,才能正式解锁。所以 400 的系统处理,在逻辑上看还是很严谨的,不会出现 UNIX 上类似于 VI 编辑时,后者覆盖前者的问题,当然这个比方打得可能有点不合适,总之就是数据的处理在逻辑上不会出现混乱,在排障时,不用在这方面想得太过复杂。

## 9.5 SAVF 的备份与恢复

SAVF, 全称 SAVE FILE, 存储文件。可以将 SAVF 视为一个存储容器, 它能够指定指定的库, 或指定的数据文件, 或源代码保存在其中, 有点类似于 UNIX 中的 TAR。

SAVF 只用于备份与恢复。虽然通常这些事情是系统管理员做的, 但是如果环境许可的情况下, 开发人员能了解这些命令, 自行做做备份, 就可以更好地对程序进行测试、保护源码。当然, 在使用 RESTORE 命令时, 一定要谨慎谨慎再谨慎, 千万不能追求操作速度, 切记切记。

### 1. 建立 SAVF

要使用 SAVF, 首先我们当然需要建立一个 SAVF。如已有自己的 SAVF, 可跳过此步。建立 SAVF 的命令是:

CRTSAVF FILE (库名/SAVF 名)

如果建立成功, 就会在指定库中, 生成一个空的 SAVF。

### 2. 清空 SAVF

使用 SAVF 前, 必须保证 SAVF 是空的。SAVF 不能追加内容。新生成的 SAVF 一定是空的, 不需要特别处理;

如果是已存在的 SAVF, 需要使用 CLRSAVF 的命令, 确保清空 SAVF

CLRSAVF FILE (库名/SAVF 名)

### 3. 将指定的库备份到 SAVF 中

备份库, 使用下列命令来进行备份

SAVLIB LIB(要备份的库名) DEV(\*SAVF) SAVF (SAVF 所在库名/SAVF 名) ACCPTH(\*YES)

ACCPTH(\*YES), 表示备份时, 备份存储路径。也就是对应于数据文件, 将其逻辑文件的相关信息也备份下来, 会增加备份时间与备份空间; 但恢复时, 不需要对逻辑文件重新建立索引, 可以省很多时间。所以在备份数据文件建议加上这个参数。当然, 如果是备份源代码, 就不需要这个参数了。

### 4. 恢复已备份的库

RSTLIB SAVLIB (备份的库名) DEV(\*SAVF) SAVF (SAVF 所在的库/SAVF 名) RSTLIB(恢复的库名)

RSTLIB 这个参数, 表示恢复的库名, 其默认值等于“ 备份的库名”。也就是说, 如果我将 FHS LIB 整个库备份下来, 再使用 RSTLIB 恢复, 如果不更改 RSTLIB 中的值, 那么将直接将 FHS LIB 整个都覆盖恢复; 而如果指定 RSTLIB 的值为 OTHERLIB, 那么将么把备份的 FHS LIB 的内容, 覆盖恢复到指定的 OTHERLIB 库中。

#### 5. 将指定的对象备份到 SAVF 中

SAV OBJ          OBJ(对象名)          LIB(对象所在的库) DEV(\*SAVF) SAVF(SAVF 所在的库/SAVF 名)

一次可以备份多个对象。

如果是备份源码，那么 OBJ 就表示源码所在的 SRCFILE，MEMBER 项就是源码名。可以使用 F4 键，来备份更多的对象，或更多的源码。

#### 6. 恢复对象

RST OBJ OBJ(\*ALL) SAV LIB(对象所在库) DEV(\*SAVF) SAVF(SAVF 所在库名/SAVF 名) RST LIB(恢复的库名)

与 RST LIB 类似。当然，OBJ 选项使用 \*ALL，表示恢复 SAVF 中备份的所有的对象，也可以指定只恢复单个/多个对象。

### 9.6 数据导入导出图解

在测试中，经常需要把数据文件中的数据导到本地，保存成 EXCEL 文件。然后通过编辑，在导入到 AS400 中，作为测试数据。或者通过 EXCEL 便利的函数等功能，统计，观察实际运行数据与理论结果数据的差异，从中发现程序的错误。有的项目也要求，用导出的运行结果数据作成单体测试报告，作为程序编码的结束。

#### 9.6.1 数据导出

下面，我们将按照实例，讲解一个数据导出的详细过程和操作。

例如，我们要导出 TST010 库下的 SYMAST 表中的数据到 E:/下，并保存为 SYMAST.ELS 文件。

要求：导出的数据符合 性别=女，且数据按照入社日期倒序排列。

首先，我们点击 PCOMM 会话工具栏的导出键，如图标注的按钮：





调出 PCOMM 集成的数据传输工具，如下图：



在此处，通常我们已经知道导出文件的库，所以在路径框中直接按提示给的格式写入数据库文件所在的路径即可。如 TST010/SYMAST。

### 抽出条件

如果对要抽出的数据有抽出，显示，排序等条件限制，则点击高级按钮，在出现的窗体中按提示配置。如下图：



The dialog box is titled "AS/400 ---> PC 传送 - 高级". It contains several input fields for query conditions, each with a "浏览..." (Browse...) button to its right:

- JOIN BY (J):
- GROUP BY (G):
- SELECT (S): \*ALL
- WHERE (W): SYSYBT = '女'
- HAVING (A):
- ORDER BY (O): SYNYDT DESC

At the bottom, there is a checkbox labeled "返回丢失字段的记录 (F):" (Return records of missing fields) and four buttons: "确定" (OK), "取消" (Cancel), "帮助 (H)" (Help), and "分组函数 (R)" (Group Functions).

### 保存格式

对导出数据的格式定义。点击第二个高级，选择如下图选项：



This dialog box shows the "输出 PC 文件" (Output PC File) section. It includes the following options:

- ☒ 替换旧文件 (R) (Replace old file)
- PC 文件类型 (T): BIFF (selected in the dropdown)
- ☒ 保存传送描述 (S) (Save transfer description)
- 描述文件名 (D): C:\PROGRAM FILES\PERSONAL COMMU (partial path visible)

At the bottom, there are three buttons: "确定" (OK), "取消" (Cancel), and "帮助 (H)" (Help).

点击确定。

### 保存路径

点击 PC 后边的浏览, 配置如图：





确定。

配置好相关参数后，点击运行，即可按条件导出 SYMAST 数据文件的数据。同时生成导出数据的描述文件 SYMAST.FDF。

#### 9.6.2 数据导入

数据的导入，一般都在对导出的数据编辑之后，且要保证导出数据的描述文件.FDF 存在并完好。在数据编辑时，一定要注意，不要改变数据的格式，在保存时候，也要保留已有的格式，否则，会因为数据的格式不正确而导致导入不成功。

另外，数据的导入，也不像导出那样具有可选择性，而是完全替换原有的数据。为了防止 AS400 中数据文件中数据的丢失，往往都导入到对应的中间数据表中，然后通过 CPYF 等数据常用命令，COPY 到数据文件中。

下面，我们也按照实例，讲解一个数据导入的详细过程和操作。

例如，我们要将 E: /下编辑好的数据 SYMAST.XLS 导回到 TST010 库下的 SYMAST 表中。

点击导出窗体的切换到发送，或会话工具栏接受按钮左边的发送按钮，都可以调出发送窗体。配置如下图：



点击运行即可。

## 第十章 附录

### 10.1 CL 程序实例:

#### 10.1.1 起动所用的初始程序（程序员）

```
PGM
CHGLIBL LIBL(TESTLIB QGPL QTEMP)
CHGJOB OUTQ(WSPRTR)
TFRCTL QPGMMENU
ENDPGM
```

测试库放在库列表的第一位，打印机选择一个输出队列且显示程序员菜单。

#### 10.1.2 把对象从测试库移到产品库中

```
PGM PARM(&OBJ &OBJTYPE &OPER)/* 对象名、类型和操作码*/
DCL &OBJ *CHAR LEN(10)
DCL &OBJTYPE *CHAR LEN(7)
DCL &OPER *CHAR LEN(1) /* R=Replace M=Move */
IF ((&OPER *NE 'M') *AND (&OPER *NE 'R')) THEN(DO)
    SNDPGMMSG MSG('Operation code must be "R" or "M" ')
    RETURN
ENDDO

IF ((&OBJTYPE *NE *PGM) *AND (&OBJTYPE *NE *FILE) *AND (&OBJTYPE +
*NE *DTAARA)) THEN(DO)
    SNDPGMMSG MSG('Object' *BCAT &OBJ *BCAT ' must be *PGM, +
*FILE, or *DTAARA')
    RETURN
ENDDO

CHKOBJ BLDLIB/&OBJ OBJTYPE(&OBJTYPE)
MONMSG MSGID(CPF9801) EXEC(DO)
    SNDPGMMSG MSG('Object or object type does not exist +
in BLDLIB')
    RETURN
ENDDO

IF (&OPER *EQ 'M') THEN(DO)
    MOVOBJ BLDLIB/&OBJ OBJTYPE(&OBJTYPE) TOLIB(PRODLIB)
    MONMSG MSGID(CPF3208) EXEC(DO)
        SNDPGMMSG MSG('Object' *BCAT &OBJ *BCAT ' +
already exists in PRODLIB')
```

```
                RETURN
                ENDDO
        CHKOBJ PRODLIB/&OBJ OBJTYPE(&OBJTYPE)
        MONMSG MSGID(CPF9801) EXEC(DO)
        SNDPGMMSG MSG('Object or object type does not +
                        exist in PRODLIB')
        RETURN
        ENDDO
    ENDDO
RETURN
ENDPGM
```

对象名、类型和操作码是从另外的过程和程序传来的。要检查操作码和对象类型是否正确，对象是否在测试库中存在。如果对象库中没有此对象，则把它从测试库移到对象库中，然后确认移动，可对移动命令加一些对象的权限，或处理其它请求和其它类型对象。

#### 10.1.3 在应用程序中保存规定的对象

```
PGM
SAVOBJ OBJ(FILE1 FILE2) LIB(LIBA) OBJTYPE(*FILE) DEV(TAP01) +
    CLEAR(*YES)
SAVOBJ OBJ(DTAARA1) LIB(LIBA) OBJTYPE(*DTAARA) DEV(TAP01)
SNDPGMMSG MSG('Save of daily backup of LIBA completed') +
    MSGTYPE(*COMP)
ENDPGM
```

这个程序保证有规律的重复过程有一致的命令入口，也可加一个 SAVOBJ 命令。这个程序依赖于操作员选择正确的软盘或磁带来为每个应用程序做周期的备份。也可以由对每个保存操作分配唯一的软盘或磁带来控制。如果要在每周分别保存工资文件，可以给不同的软盘或磁带名，然后写一个程序来比较软盘或磁带名是否与这周用的相同。

#### 10.1.4 提交作业（系统操作员）

```
PGM /*DAILYAC*/
SBMJOB JOB(DAILYACCRC) JOBD(ACCRC2) +
    CMD(CALL ACCRC305 PARM(DAILY))
SNDPGMMSG MSG('Daily Accounts Receivable job DAILYACCRC +
    submitted to batch') MSGTYPE(*COMP)
ENDPGM
```

系统操作员不用填写提交作业的所有参数，只调用此程序既可。

## 10.1.5 打印程序输出

```
PGM
DCL          VAR(&PRT) TYPE(*CHAR) LEN(10)
/*打印机指定 */
OVRPRTF      FILE(BAIQ310P) PAGESIZE(60 120) LPI(6)  +
              CPI(10) OVRFLW(60) OUTQ(PRT01)

/* 运行打印 RPG 程序*/
CALL          PGM(BAIQ310R)

END:          RCLRSC
              ENDPGM
OVRPRTF 编辑报表的属性 OUTQ(PRT01)要输出到的打印机 PRT01 为打印机名
```

## 10.1.6 在 QTEMP 下生成临时文件

```
PGM
/*  */
DCL          VAR(&WKLIBF) TYPE(*CHAR) LEN(10)

/*检查 QTEMP 下临时文件 WKIQ40 是否存在*/
CHKOBJ       OBJ(QTEMP/WKIQ40) OBJTYPE(*FILE)
MONMSG       MSGID(CPF9801) EXEC(DO)

/* RTVOBJD 查找 JOB 属性 */
RTVOBJD      OBJ(WKIQ40) OBJTYPE(*FILE) RTNLIB(&WKLIBF)

/* CRTDUPOBJ 创建复制 OBJ 从某库到 QTEMP*/
CRTDUPOBJ    OBJ(WKIQ40) FROMLIB(&WKLIBF) +
              OBJTYPE(*FILE) TOLIB(QTEMP)
ENDDO

/*初始化临时文件*/
CLRPFM       FILE(QTEMP/WKIQ40)
/*  */
MONMSG       MSGID(CPF0000)

/*复写物理文件 */
OVRDBF       FILE(WKIQ40) TOFILE(QTEMP/WKIQ40)

/*运行使用临时文件的程序*/
CALL          PGM(BAIQ400R)

/*删除 QTEMP 下的临时文件 WKIQ40 */
DLTF         FILE(QTEMP/WKIQ40)

END:          RCLRSC
              ENDPGM
```

## 10.2 AS/400 常用 CL 命令表

1	ADDJOBSCDE	添加批处理
2	ADDLFM	DB 逻辑文件添加 MEMBER
3	ADDLIB	追加 LIB 到现有的编译 LIB 中
4	ADDMSGD	添加消息
5	ADDPFM	DB 物理文件添加 MEMBER
6	ADDTCPIFC	添加 TCP/IP 接口
7	ALCOBJ	为 OBJ 分配内存
8	CALL	用来调用其他程序
9	CHGCURLIB	修改当前库
10	CHGDEVDS	修改装置设备属性
11	CHGDTAARA	DATA 域变更
12	CHGJOB	改变 JOB 的属性
13	CHGJOBSCDE	变更批处理
14	CHGLIB	修改库的属性
15	CHGLIBL	变更编译时的库
16	CHGOBJD	修改 OBJ 文件属性描述
17	CHGOBJOWN	OBJ 所有者变更
18	CHGPFM	修改程序的相关信息
19	CHGPGM	修改 PGM 属性
20	CHGPRTF	帐票 OBJ 属性变更
21	CHGPWD	修改密码
22	CHGSAVF	SAVE 文件变更
23	CHGSPLFA	变更 TAPE 或帐票的属性
24	CHGSRCPF	修改文件夹的属性
25	CHGTCPIFC	变更 TCP/IP 接口
26	CHGUSRPRF	用户变更(修改密码等)
27	CHGVAR	CL 中改变变量值
28	CHKOBJ	查找 OBJ
29	CHKPRDOPT	检查产品选项
30	CLRLIB	清除库中的内容
31	CLRROUTQ	清除输出队列
32	CLRPFM	清除物理文件数据
33	CLRSVF	删除 SAVE 文件下的内容
34	CPYF	复制文件
35	CPYFRMQRYF	复制 QUERY 文件到目的库
36	CPYLIB	库复制

37	CPYSPLF	把 CALL 出来的帐票文件导成数据文件
38	CPYSRCF	复制原文件
39	CPYTOIMPF	导入复制
40	CRTAPW	挂线程序做成
41	CRTBNDCL	编译 CLLE 文件
42	CRTBNDRPG	编译 RPGLE 程序
43	CRTCLPGM	编译 CL 文件
44	CRTDEVDSF	创建终端装置设备
45	CRTDEVPRT	创建打印设备
46	CRTDSPF	编译画面文件
47	CRTDTAARA	创建 DATA 域
48	CRTDUPOBJ	创建复制 OBJ 从某库到某库
49	CRTJRN	创建日志
50	CRTJRNRCV	创建日志接受器
51	CRTL F	逻辑文件编译
52	CRTL I B	创建一个 LIB
53	CRTL I NETH	设置线
54	CRTMSGF	创建消息文件
55	CRTPF	物理文件编译
56	CRTRPGPGM	编译 RPG 程序
57	CRTSAVF	创建一个 SAVE 文件
58	CRTSRCF	创建一个文件夹
59	CRTSRVPGM	创建服务程序
60	CRTUSRPRF	新建用户
61	CVTDAT	变更 JOB 的时间格式
62	CVTRPGSRC	代码转换
63	DLTF	删除 OBJ 文件
64	DLTJRN	删除日志
65	DLTJRNRCV	删除日志接受器
66	DLTL I B	删除库
67	DLTL I CPGM	删除特许程序
68	DLTMSGF	删除消息文件
69	DLTPGM	删除程序文件
70	DLTQRY	QUERY 删除
71	DLTUSRPRF	删除用户
72	DOWNLOADSAVF	下载 SAVE 文件
73	DSPDBR	显示数据文件关联

74	DSPDTAARA	查看域的内容
75	DSPFD	查看文件信息
76	DSPFFD	查看 DB-OBJ 的字段
77	DSPJOBLOG	查看 JOB 的日志
78	DSPLIB	库表示
79	DSPMSG	查看 MSG
80	DSPOBJD	查看 OBJ 的属性或查看 DB-OBJ 的 SRC
81	DSPPFM	查看物理文件的数据
82	DSPPGM	查看 PGM 的信息
83	DSPUSRPRF	显示用户信息
84	EDTF	文件编辑
85	EDTLIBL	修改编译是的库
86	ENDJOB	结束一个活动的 JOB
87	ENDJRNP	解除日志和 PF 文件的关联
88	FNDSTRPDM	查找指定的字符串在指定的文件组中
89	GO	菜单表示
90	GO LICPGM	查看 AS400 安装了那些软件包(MENU)
91	MONMSG	信息监测
92	MOVOBJ	移动 OBJ 文件
93	MARGPW	合成挂线
94	NETSTAT	网络状况处理
95	OPNDBF	打开物理文件
96	OPNQRYF	打开 QUERY 文件
97	OVRDBF	复写物理文件
98	RGZPFM	清源代码后面的时间压缩等
99	RMVJRNCHG	恢复记录
100	RMVLIBL	从库列表中移除
101	RMVM	删除文件
102	RNMM	文件重命名
103	RNMOBJ	OBJ 重命名
104	RSTLIB	恢复库
105	RSTLICPGM	复原特许程序
106	RSTOBJ	复原 SAV 文件
107	RTVCLSRC	CL 查找
108	RTVDTAARA	DATA 域属性检索
109	RTVJOBA	查找 JOB 属性
110	RTVMBRD	OBJ 文件记述检索



111	RTV0BJD	OBJ 参照
112	RTVUSRPRF	用户文件属性检索
113	RUNQRY	运行一个 QUERY
114	SAV	保存数据库目录
115	SAVLIB	库的保存
116	SAVLCPGM	保存特许程序
117	SAV0BJ	保存文件
118	SAVSAVFTA	把 SAVE 文件备份到磁盘上
119	SBMJ0B	JOB 投入
120	SND BRKMSG	发送消息到指定的用户
121	STRDBG	调试 RPGLE 程序并设置断点
122	STRISDB	调试 RPG 程序并设置断点
123	STRJRNPF	启动日志
124	STRPRTWTR	启动打印机
125	STRRLU	预览 PRT 帐票文件
126	STRSDA	查看画面文件
127	STRSEU	编写代码
128	UPDDTA	修改数据
129	UPSAVF	上传 SAVE 文件
130	VRFCFG	把锁死的终端用户激活
131	WRKDEVD	显示各种用户
132	WRKHDWRSC	查看硬件资源
133	WRKJOBSCDE	查看批处理
134	WRKLIBPDM	PDM 工作目录
135	WRKLIBINF	查看 AS400 安装了那些软件包(COMMAND)
136	WRKM BRPDM	PDM 成员处理
137	WRK0BJ	查看 OBJ 文件都在那些库下有
138	WRK0BJLCK	查看 OBJ 文件谁在用
139	WRK0BJPDM	PDM 的 OBJ 处理
140	WRKQRY	QUERY 处理
141	WRKSPLF	查看打印池状态
142	WRKSYSACT	查看系统活动 JOB
143	WRKSYSVAL	系统参数的变更
144	WRKTCPSTS	用 TCP/IP 查看 AS/400 用户占用时间
145	WRKUSRJOB	用户下的 JOB 处理
146	WRKUSRPRF	用户的增删改
147	WRKWTR	查看打印机状态

### 10.3 RPG 程序实例：

#### 10.3.1 查询（指示画面+SFL+报表）

##### DSP 文件源码

```

A*****
A*
A* FILE ID      BAIQ310D
A* FILE NAME    开箱前空货位查询画面
A*
A* -----
A* CREATION DATE      08-02-18
A* UPDATION DATE      -  -
A*
A*****
A                                DSPSIZ(24 80 *DS3)
A                                PRINT
A                                CF03
A                                CF12
A*****
A*          第一画面
A*****
A          R FMT01
A
A                                OVERLAY
A                                1 2' BAIQ310D'
A                                COLOR(BLU)
A                                1 12' 01'
A                                COLOR(BLU)
A                                1 26' ** 到货零件空货位查询 **'
A                                DSPATR(RI)
A                                DSPATR(HI)
A                                1 57DATE
A                                EDTCDE(Y)
A                                COLOR(BLU)
A                                1 66TIME
A                                COLOR(BLU)
A                                D1WSID      5A 0 1 75COLOR(BLU)
A*
A                                2 2' 仓库代码: '
A                                COLOR(BLU)
A                                D1WHCD      3A B 2 14TEXT('仓库代码')
A                                DSPATR(CS)
A                                DSPATR(UL)
A 40                                DSPATR(RI)
A 40                                DSPATR(PC)
A*
A                                2 62' W/H'
A                                COLOR(BLU)
A                                D1WHC1      3A 0 2 66TEXT('WAREHOUSE CODE')
A                                DSPATR(HI)
A                                D1USID      10A 0 2 70TEXT('USE ID')
A                                DSPATR(HI)
A*
A                                3 2' 发票号码: '
A                                COLOR(BLU)
A                                D1SPNO      12A B 3 14TEXT('发票号码')
A                                DSPATR(CS)

```



```
A
A 41 DSPATR(UL)
A 41 DSPATR(RI)
A 41 DSPATR(PC)
A 3 28' 状态:'
A COLOR(BLU)
A D1FLST 1A B 3 36TEXT(' 状态' )
A DSPATR(CS)
A DSPATR(UL)
A 42 DSPATR(RI)
A 42 DSPATR(PC)
A 3 39' (1 开箱前 2 开箱后) '
A*****
A* 第二画面 S F L
A*****
A*
A R SFL02 SFL
A*
A S2SEQ 4Y 00 6 2TEXT(' SEQ' )
A EDTCDE(3)
A DSPATR(HI)
A S2SPNO 12A 0 6 7TEXT(' 发票号码' )
A DSPATR(HI)
A S2VACD 2A 0 6 20TEXT(' VAN' )
A DSPATR(HI)
A S2FLNO 8A 0 6 23TEXT(' FLOAT' )
A DSPATR(HI)
A S2LNNO 5Y 00 6 32TEXT(' 行号' )
A EDTCDE(3)
A DSPATR(HI)
A S2NPS 15A 0 6 38TEXT(' 零件号' )
A DSPATR(HI)
A S2FLQY 7Y 00 6 54TEXT(' 数量' )
A EDTCDE(3)
A DSPATR(HI)
A S2FLST 1A 0 6 63TEXT(' 状态' )
A DSPATR(HI)
A S2VDCD 9A 0 6 67TEXT(' 供应商' )
A DSPATR(HI)
A S2WHCD 3A 0 6 77TEXT(' 仓库' )
A DSPATR(HI)
A*****
A* 第二画面 C T L
A*****
A*
A R CTL02 SFLCTL(SFL02)
A SFLSI Z(9999)
A SFLPAG(0016)
A OVERLAY
A PAGEUP(36 ' PAGEUP' )
A PAGEDOWN(37 ' PAGEDOWN' )
A 43 CF06
A 32 SFLCLR
A 33 SFLDSPCTL
A 34 SFLDSP
A* 35 SFLEND
A D2RNBR 4S 0H SFLRCDNBR
```

```

A*
A      1  2' BAI Q310D'
A      COLOR(BLU)
A      1 12' 02'
A      COLOR(BLU)
A      1 26' ** 到货零件空货位查询 **'
A      DSPATR(RI)
A      DSPATR(HI)
A      1 57DATE
A      EDTCDE(Y)
A      COLOR(BLU)
A      1 66TIME
A      COLOR(BLU)
A      D1WSID      5A  0  1 75COLOR(BLU)
A*
A      2  2' 仓库代码:'
A      COLOR(BLU)
A      D1WHCD      3A  0  2 14TEXT('仓库代码')
A      DSPATR(HI)
A*
A      2 62' W/H'
A      COLOR(BLU)
A      D1WHC1      3A  0  2 66TEXT('WAREHOUSE CODE')
A      DSPATR(HI)
A      D1USID      10A  0  2 70TEXT('USE ID')
A      DSPATR(HI)
A*
A      3  2' 发票号码:'
A      COLOR(BLU)
A      D1SPNO      12A  0  3 14TEXT('发票号码')
A      DSPATR(HI)
A      3 28' 状态:'
A      COLOR(BLU)
A      D1FLST      1A  0  3 36TEXT('状态')
A      DSPATR(HI)
A      3 38' (1 开箱前 2 开箱后) '
A*SFL 标题
A      5  2' SEQ 发票号码  VAN FLOAT  行+
A      号零件号      数量状态+
A      供应商仓库'
A      DSPATR(UL)
A      COLOR(BLU)
A      3 61' 行数:'
A      COLOR(BLU)
A      4 61' 数量:'
A      COLOR(BLU)
A      D2TTLN      7Y  00  3 72TEXT('行数')
A      EDTCDE(3)
A      DSPATR(HI)
A      D2TTQY      10Y  00  4 69TEXT('数量')
A      EDTCDE(3)
A      DSPATR(HI)
A*****
A*      功能键&错误信息
A*****
A      R FMT99

```



```
A**
A          D9CMD          780  0 23  2TEXT('CMD' )
A                                COLOR(BLU)
A          D9EMSG          780  0 24  2TEXT('ERROR MSG' )
A                                DSPATR(HI)
A*****
```

## PRT 文件源代码

```
A*****
A          R HDR
A*****
A          HDPGID          10A  0  2  3TEXT('PGM ID' )
A                                2 22' 到货零件空货位清单'
A*                                CHRSLZ(2 2)
A                                2 53' DATE: '
A          HDDATE          8S  00  2 58
A                                EDTWRD('    /    /    ')
A          HDWHCD          3A  0  3  9
A                                3 1' 仓库: '
A                                3 53' PAGE: '
A                                3 58PAGNBR
A                                EDTCDE(3)
A                                4 2' SEQ 发票号码'
A                                4 20' VAN FLOAT'
A                                4 33' 行号零件号'
A                                4 58' 数量状态供应商'
A*****
A          R DTL
A*****
A          DTSEQ          3S  00
A                                SPACEB(001)
A                                2TEXT('SEQ' )
A                                EDTCDE(3)
A          DTSPNO          12A  0
A                                7TEXT(' 发票号码' )
A          DTVACD          2A  0
A                                20TEXT(' VAN 号' )
A          DTFLNO          8A  0
A                                24TEXT(' FLOAT 号' )
A          DTLNNO          5S  00
A                                33TEXT(' FLOAT 行号' )
A                                EDTCDE(3)
A          DTNPS          15A  0
A                                40TEXT(' 零件号' )
A          DTFLQY          7S  00
A                                56TEXT(' 数量' )
A                                EDTCDE(3)
A          DTFLST          1A  0
A                                66TEXT(' 开箱状态' )
A          DTVDCD          9A  0
A                                71TEXT(' 供应商' )
A*****
```



## RPG 文件源代码

```
H          Y/
H*****
H*
H*   PROGRAM TITLE - 开箱前空货位查询
H*       ID      -  BAI Q310R
H*
H*   DESCRIPTION   -  根据入力仓库，发票号和开箱状态从收货头
H*                   文件和收货明细文件里查询数据。
H*                   查询结果打印。
H*   PROGRAMMED BY - XXXXXXXX      DATE 08/02/18
H*   MODIFIED   BY - X.XXXXXXXX    DATE 00/00/00
H*   MODIFIED   BY - X.XXXXXXXX    DATE 00/00/00
H*
H*   COPYRIGHT 2006 HONDA MOTOR CO.,LTD.
H*
H*****
H*   INTERNAL SUBROUTINE USAGE
H*-----
H*   @INZ  画面初期化(1.)
H*   @FMT01 第一画面(2-1.)
H*   @INZ1 第一画面初期化(2-1-1.)
H*   @CHK1 第一画面CHECK(2-1-2.)
H*   @FMT02 第二画面(2-2.)
H*   @INZ2 第二画面初期化(2-2-1.)
H*   @SET  设定出力数据读取位置
H*   @PGUP 向上读取(2-2-2.)
H*   @PGUP1 向上读取1(2-2-2.)
H*   @PGUP2 向上读取2(2-2-2.)
H*   @PGDN 向下读取(2-2-2.)
H*   @RED1 收货明细文件(FLDMSTL9)读取(2-2-2.)
H*   @SFAD1 SUBFILE EDIT(4.)
H*   @PRT01 印刷处理(2-2-4.)
H*   @SETD  SET CURRENT DATE FORMAT
H*   @PRT  印刷数据出力(2-2-4-1.)
H*   @RST  ERROR RESET
H*   @MSG  DISPLAY ERROR MESSAGE
H*   @RST  ERROR MESSAGE CLEAR
H*   @END  END PROGRAM
H*   @DEF  DEFINITION OF FIELD
H*****
H*   INDICATOR USAGE
H*-----
H*   32      SUBFILE CLEAR          INDICATOR
H*   33      SUBFILE DISPLAY CONTROL INDICATOR
H*   34      SUBFILE DISPLAY        INDICATOR
H*   35      SUBFILE END            INDICATOR
H*   36      ROLL UP                INDICATOR
H*   37      ROLL DOWN              INDICATOR
H*   43      CF06                   INDICATOR
H*   99      TOTAL ERROR            INDICATOR
H*   90      CHAIN                  INDICATOR
H*   94      READE FLDMSTL9         INDICATOR
F*****
F*   FILE DESCRIPTION
F*-----
```



```
F*   BAIQ310D   DISPLAY FILE                TYPE(DSPF)  *
F*   SCTBL3X1   系统控制表                  TYPE(INDEX) *
F*   FLHMSTX6   收货头文件                  TYPE(INDEX) *
F*   FLDMSTL9   收货明细文件                TYPE(INDEX) *
F*   BAIQ310P   PRINT FILE                  TYPE(PRTF)  *
F*   -----*
F*****
F*           FILE DESCRIPTION                *
F*****
F*<开箱前空货位查询画面文件>
FBAIQ310DCF   E                               WORKSTN
F                                           KINFDS INFDS
F                                           W#RRN1KSFILE SFL02
F*<系统控制表>
FSCTBL3X1IF   E           K           DISK
F*<收货头文件>
FFLHMSTX6IF   E           K           DISK
F*<收货明细文件>
FFLDMSTL9IF   E           K           DISK
F*<零件空货位清单>
FBAIQ310PO    E           88          PRINTER
E*****
E*           ARRAY STRUCTURE                *
E*****
E           #CMD      1  2  78              COMMAND KEY
E*****
I*           DATA STRUCTURE                *
I*****
I           SDS
I
I           1  10 W#PGID
I           244 253 W#WSID
I           254 263 W#USID
I           DS
I
I           1  140#TMDT
I           1  60#TIME
I           7  140#DATE
I           7  80#DATD
I           9  100#DATM
I           11 140#DATY
I*<<系统时间取得>>
I           DS
I
I           1  80#DAT2
I           1  40#DTY2
I           5  60#DTM2
I           7  80#DTD2
IINFDS      DS
I
I           B 370 3710CSRLOC
I           B 378 3790W#RRN
C*****
C**           MAIN ROUTINE                *
C*****
C*
C           DO      *HIVAL
C           #CTL    CASEQ*BLANK      @INZ      INITIALIZE
C           #CTL    CASEQ'FMT01'     @FMT01     FIRST SCREEN CONTROL
C           #CTL    CASEQ'FMT02'     @FMT02     SECOND SCREEN CONTROL
```



```
C      #CTL      CASEQ' PRT01'   @PRT01      PRINT SCREEN CONTROL
C      #CTL      CASEQ' END   '   @END          PROGRAM END
C
C*
C      ENDDO
C*****
C*      @INZ      画面初期化      *
C*****
C      @INZ      BEGSR
C**
C      TIME      #TMDT
C      MOVEW#WSID D1WSID      WS-ID
C      MOVEW#USID D1USID      US-ID
C*取得默认仓库
C      CALL 'SPGETWHR'
C      PARM      P#WHCD      WHAREHOUSER CODE
C      PARM      P#RFLG      RETURN FLAG
C*返回值='1' 时
C      P#RFLG      IFEQ '1'
C      MOVEW'END '   #CTL      PROGRAM END
C      ELSE
C*返回值='1' 以外时
C      MOVEW'FMT01'   #CTL      FIRST SCREEN CONTROL
C      MOVEW#WHCD     D1WHCD     WHAREHOUSER CODE
C      MOVEW#WHCD     D1WHC1     P      WAREHOUSE CODE
C      ENDIF
C**
C      ENDSR
C*****
C*      @FMT01    第一画面      *
C*****
C      @FMT01    BEGSR
C**
C*第一画面初期化
C      EXSR @INZ1
C*
C      W#LOP1    DOWEQ'1'
C      MOVEW#CMD,1 D9CMD      COMMAND KEY
C      WRITEFMT99
C      SETON      33
C      EXFMTFMT01
C      SETOF      33
C*
C      SELEC
C*<<PF03 OR PF12>>
C      *INKL      WHEQ *ON      PROGRAM END
C      *INKC      OREQ *ON      PROGRAM END
C      MOVEW'END '   #CTL      CONTROL
C      MOVEW'O'      W#LOP1     LOOP FLAG
C*<<ENTER>>
C      OTHER
C      EXSR @CHK1
C      N99        MOVEW'FMT02'   #CTL      FIRST SCREEN CHECK
C      N99        MOVEW'O'      W#LOP1     SECOND SCREEN
C      ENDSL      LOOP FLAG
C      ENDDO
```



```

C**
C          ENDSR
C*****
C*          @INZ1    第一画面初期化          *
C*****
C          @INZ1    BEGSR
C**
C*显示打印信息
C          *INKF      I FEQ *ON
C                      MOVE' MSG60064' #MSGID      MESSAGE ID
C                      SETON                          99
C                      EXSR @MSG
C                      ELSE
C                      EXSR @RST
C                      ENDIF
C                      MOVE' 1'          W#LOP1          LOOP1 FLAG
C**
C          ENDSR
C*****
C*          @CHK1    第一画面 C H E C K          *
C*****
C          @CHK1    BEGSR
C**
C*MESSAGE CLEAR
C                      EXSR @RST
C                      SETOF                          99
C*仓库代码必须输入
C          D1WHCD      I FEQ *BLANK          仓库代码
C          N99          MOVE' MSG60184' #MSGID      MESSAGE ID
C                      SETON                          9940
C                      GOTO #CKED1
C                      ENDIF
C*仓库代码 SCTBL3 中存在检查
C                      MOVE' WRHS'      K#CNST      P      CNST
C                      MOVE' D1WHCD      K#SKEY      P      SKEY
C                      MOVE' A'          K#ATFG      ATFG
C          KEYC31      CHAIN SCTBL3          90
C          *IN90      I FEQ *ON
C          N99          MOVE' MSG60067' #MSGID      MESSAGE ID
C                      SETON                          9940
C                      GOTO #CKED1
C                      ENDIF
C*
C*发票号码发货头文件中存在检查
C          D1SPNO      IFNE *BLANK
C                      MOVE' D1SPNO      K#SPNO      P      发票号
C          KEYFH1      CHAIN FLHMST          90
C          *IN90      I FEQ *ON
C          FHATFG      ORNE 'A'
C          N99          MOVE' MSG60223' #MSGID      MESSAGE ID
C                      SETON                          994041
C                      GOTO #CKED1
C                      ENDIF
C                      ENDIF
C*开箱状态检查
C          D1FLST      IFNE *BLANK          开箱状态

```



```
C          D1FLST  ANDNE' 1'          开箱状态
C          D1FLST  ANDNE' 2'          开箱状态
C  N99          MOVE' MSG60213' #MSGID  MESSAGE ID
C          SETON          9942
C          GOTO #CKED1
C          ENDIF
C*
C          #CKED1  TAG
C  99          EXSR @MSG
C**
C          ENDSR
C*****
C**          @FMT02  第二画面          *
C*****
C          @FMT02  BEGSR
C**
C*第二画面初期化
C          EXSR @INZ2
C*
C          W#LOP2  DOWEQ' 1'          LOOP FLAG
C*<<COMMAND KEY>>
C          W#RRN1  MOVE' #CMD, 2  D9CMD  COMMAND KEY
C          IFNE *ZERO
C          SETON          34
C          ELSE
C          SETOF          34
C          ENDIF
C*<<SCREEN DI SPLAY>>
C          WRITEFMT99
C          SETON          33
C          EXFMTCTL02
C          SETOF          33
C          Z-ADDW#RRN  D2RNBR          SFL CURSOR
C*
C          SELEC
C*<<PAGE UP>>
C          *IN36  WHEQ *ON          PAGE UP
C*<<PAGE DOWN>>
C          *IN37  WHEQ *ON          PAGE DOWN
C*<<PF03>>
C          *INKC  WHEQ *ON          PF03: END
C          MOVE' END '  #CTL
C          MOVE' O'  W#LOP2
C*<<PF06>>
C          *INKF  WHEQ *ON          PF06: PRINT
C          MOVE' PRT01'  #CTL
C          MOVE' O'  W#LOP2
C*<<PF12>>
C          *INKL  WHEQ *ON          PF12: FRONT SCREEN
C          MOVE' FMT01'  #CTL
C          MOVE' O'  W#LOP2
C*<<ENTER>>
C          OTHER          ENTER
C          ENDSL
C*没有明细数据时设定打印（ P F 0 6 ）
C          W#RRNN  IFEQ *ZERO
```



```
C      SETOF      43      PF06
C      MOVEL' MSG00009' #MSGID
C      EXSR @MSG
C      ELSE
C      SETON      43      PF06
C      ENDIF
C      ENDDO
C**
C      ENDSR
C*****
C*      @INZ2      第二画面初期化      *
C*****
C      @INZ2      BEGSR
C**
C* CLEAR
C      EXSR @RST
C      MOVEL *BLANK      W#RED1      READ FLAG
C      MOVEL '1'      W#LOP2      LOOP FLAG
C      Z-ADD *ZERO      D2TTLN
C      Z-ADD *ZERO      D2TTQY
C* 初始化第二画面
C      SETON      32
C      WRITECTLO2
C      SETOF      32
C*
C      Z-ADD *ZERO      W#RRNN      SFL RRN
C      Z-ADD *ZERO      W#RRN1      SFL RRN
C      Z-ADD *ZERO      W#SEQ      SEQ
C* 设定出力数据读取位置
C      EXSR @SET
C      EXSR @RED1
C* 出力明细数据
C      N91      EXSR @SFAD1
C*
C* 没有明细数据时设定打印（P F 0 6）
C      W#RRNN      IFEQ *ZERO
C      SETOF      43      PF06
C      MOVEL' MSG00009' #MSGID
C      EXSR @MSG
C      ELSE
C      SETON      43      PF06
C      ENDIF
C**
C      ENDSR
C*****
C*      @SET      设定出力数据读取位置      *
C*****
C      @SET      BEGSR
C**
C      SETOF      50
C* 发票号入力判断
C      D1SPNO      IFNE *BLANK      发票号入力时
C      MOVELD1WHCD      K#WHCD      仓库
C      MOVEL *BLANK      K#LOC1      货位
C      MOVELD1SPNO      K#SPNO      P      发票号
C      KEYFD2      SETLLFLDMST
```



```
C          SETOF          50
C          ELSE
C          MOVELD1WHCD    K#WHCD    P    发票号没入力时
C          MOVEL*BLANK    K#LOC1    仓库
C          KEYFD1        SETLLFLDMST  货位
C          SETON          50
C          ENDIF
C**
C          ENDSR
C*****
C*          @RED1    收货明细文件(FLDMSTL9)读取          *
C*****
C          @RED1    BEGSR
C**
C          MOVEL*BLANK    W#RED1    READ FLAG
C          W#RED1    DOUNE*BLANK
C*读取发货明细文件记录
C   N50    KEYFD2    REAEFLDMST    91
C   50    KEYFD1    REAEFLDMST    91
C          *IN91    IFEQ *OFF
C*
C*画面开箱状态: 1(开箱前)
C          D1FLST    IFEQ '1'
C          FDFLST    ANDGE'20'
C          ITER
C          ENDIF
C*
C*画面开箱状态: 2(开箱后)
C          D1FLST    IFEQ '2'
C          FDFLST    ANDLT'20'
C          ITER
C          ENDIF
C*
C          MOVEL'O'    W#RED1    READ FLAG
C          ELSE
C          MOVEL'1'    W#RED1    READ FLAG
C          ENDIF
C*
C          ENDDO
C**
C          ENDSR
C*****
C*          @SFAD1    SUBFILE EDIT          *
C*****
C          @SFAD1    BEGSR
C**
C          W#RRNN    DOUGE9999
C*
C          W#RED1    IFEQ '1'    当前没有数据, 且
C          LEAVE    结束
C          ENDIF
C*<<SUBFILE CLEAR>>
C          CLEARSFLO2
C*<<RRN SET>>
C          ADD 1    W#SEQ    SEQ
C          ADD 1    W#RRNN    SFL RRN
```



```
C
C*<<SUBFILE EDIT>>
C      W#SEQ      I FLE 9999
C      Z-ADDW#SEQ      S2SEQ      SEQ
C      ELSE
C      Z-ADD9999      S2SEQ      SEQ
C      ENDIF
C      MOVELFDSPNO      S2SPNO      发票号码
C*
C      MOVELFDSPNO      K#SPNO      P      发票号
C      MOVELFDFLNO      K#FLNO      P      FLOAT 号
C      KEYFH2      CHAI NFLHMST      92
C      *IN92      I FEQ *OFF
C      MOVELFHVACD      S2VACD      VAN 号
C      ENDIF
C      MOVELFDFLNO      S2FLNO      FLOAT 号
C      Z-ADDFDLNNO      S2LNN0      FLOAT 行号
C      MOVELFDNPS      S2NPS      零件号
C      Z-ADDFDFLQY      S2FLQY      数量
C      MOVELFDVDCD      S2VDCD      供应商
C      MOVELFDWHCD      S2WHCD      仓库号
C      FDFLST      I FLT '20'      开箱
C      MOVEL'1'      S2FLST      开箱状态
C      ELSE
C      MOVEL'2'      S2FLST      开箱状态
C      ENDIF
C*写入S F L并读取次条
C      WRITESFLO2
C      ADD 1      D2TTLN
C      ADD FDFLQY      D2TTQY
C      EXSR @RED1
C      ENDDO
C*
C      Z-ADD1      D2RNBR      SFL RRN
C*<<SUBFILE END>>
C      W#RRN1      I FEQ 9999
C      W#RED1      OREQ '1'
C      SETON      35
C      ELSE
C      SETOF      35
C      ENDIF
C**
C      ENDSR
C*****
C*      @PRT01      印刷处理      *
C*****
C      @PRT01      BEGSR
C**
C*设定出力数据读取位置
C      EXSR @SET
C      EXSR @RED1
C*印刷数据出力
C      N91      EXSR @PRT
C*返回第一画面
C      MOVEL' FMT01'      #CTL
C**
```



```
C          ENDSR
C*****
C*          @PRT      印刷数据出力          *
C*****
C          @PRT      BEGSR
C**
C          SETON          88
C          EXSR @SETD
C          Z-ADD#DAT2      HDATE      日期
C          MOVEW#PGID      HPGID      PGID
C          MOVELD1WHCD      HDWHCD      仓库
C          CLEARDTL
C          *IN91      DOWEQ*OFF
C*帐票头部输出
C      88          WRITEHDR
C      88          SETOF          88
C          DTSEQ      IFLT 999
C          ADD 1          DTSEQ      SEQ
C          ELSE
C          Z-ADD999      DTSEQ      SEQ
C          ENDIF
C          MOVELFDSPNO      DTSPNO      发票号码
C*
C          MOVELFDSPNO      K#SPNO      P      发票号
C          MOVELFDLNO      K#FLNO      P      FLOAT 号
C          KEYFH2      CHAINFLHMST      92
C          *IN92      IFEQ *OFF
C          MOVELFHVACD      DTVACD      VAN 号
C          ENDIF
C          MOVELFDLNO      DTFLNO      FLOAT 号
C          Z-ADDFDLNNO      DTLNNO      FLOAT 行号
C          MOVELFDNPS      DTNPS      零件号
C          Z-ADDFDLQY      DTFLQY      数量
C          MOVELFDVDCD      DTVDCD      供应商
C          FDFLST      IFLT '20'      开箱
C          MOVEL'1'      DTFLST      开箱状态
C          ELSE
C          MOVEL'2'      DTFLST      开箱状态
C          ENDIF
C*帐票明细输出
C          WRITEDTL
C*读取次条
C          EXSR @RED1
C          ENDDO
C**
C          ENDSR
C*****
C*          @SETD      SET CURRENT DATE FORMAT      *
C*****
C          @SETD      BEGSR
C**
C          TIME          #TMDT          TM
C          Z-ADD#DATD      #DTD2          YY
C          Z-ADD#DATM      #DTM2          MM
C          Z-ADD#DATY      #DTY2          DD
C**
```

```

C                                ENDSR
C*****
C*          @RST      ERROR RESET                      *
C*****
C          @RST      BEGSR
C**
C                                MOVEA' 000000'  *IN, 40          40-45
C                                MOVEA' 00'      *IN, 98
C                                MOVEL *BLANK      D9EMSG
C                                MOVEL *BLANK      #MSGID
C**
C                                ENDSR
C*****
C*          @MSG      ERROR MESSAGE                      *
C*****
C          @MSG      BEGSR
C**
C                                CALL ' CMEX010R'
C                                PARM              #MSGID      ERROR ID
C                                PARM              D9EMSG       ERROR MSG
C**
C                                ENDSR
C*****
C*          @END      PROGRAM END                      *
C*****
C          @END      BEGSR
C**
C                                SETON              LR
C                                RETRN
C**
C                                ENDSR
C*****
C*          @DEF      DEFINITION OF FIELD                      *
C*****
C          @DEF      BEGSR
C*****
C* KLIST *
C*****
C*<< SCTBL3L1 KEY-LIST 1 >>
C          KEYC31      KLIST
C                                KFLD              K#CNST          CONSTANT KEY
C                                KFLD              K#SKEY          SUB KEY
C                                KFLD              K#ATFG          ACTIVE FLAG
C*<< FLHMSTX6 KEY-LIST >>
C          KEYFH1      KLIST
C                                KFLD              K#SPNO          发票号
C          KEYFH2      KLIST
C                                KFLD              K#SPNO          发票号
C                                KFLD              K#FLNO          FLOAT 号
C*<< FLDMSTL9 KEY-LIST >>
C          KEYFD1      KLIST
C                                KFLD              K#WHCD          仓库
C                                KFLD              K#LOC1          货位
C          KEYFD2      KLIST
C                                KFLD              K#WHCD          仓库
C                                KFLD              K#LOC1          货位

```



C		KFLD	K#SPNO	发票号
C*	变量定义			
C	*LIKE	DEFN C3CNST	K#CNST	CONSTANT KEY
C	*LIKE	DEFN C3SKEY	K#SKEY	SUB KEY
C	*LIKE	DEFN C3ATFG	K#ATFG	ACTIVE FLAG
C	*LIKE	DEFN FDWHCD	K#WHCD	仓库
C	*LIKE	DEFN FDSPNO	K#SPNO	发票号
C	*LIKE	DEFN FDFLNO	K#FLNO	FLOAT 号
C	*LIKE	DEFN FDLOC1	K#LOC1	货位
C*				
C	*LIKE	DEFN FDWHCD	P#WHCD	仓库
C	*LIKE	DEFN FDSPNO	W#SPNO	发票号
C	*LIKE	DEFN FDFLNO	W#FLNO	FLOAT 号
C*				
C		MOVEL*BLANK	W#LOP1 1	画面 1 LOOP FLAG
C		MOVEL*BLANK	W#LOP2 1	画面 2 LOOP FLAG
C		MOVEL*BLANK	W#RED1 1	READ FLAG2
C		MOVEL*BLANK	W#RED3 1	READ FLAG3
C		Z-ADD*ZERO	W#RRNN 40	SFL RRN
C		Z-ADD*ZERO	W#RRN1 40	SFL RRN
C		MOVEL*BLANK	P#RFLG 1	RETURN FLAG
C		MOVEL*BLANK	#MSGID 8	MESSAGE ID
C		MOVEL*BLANK	#CTL 6	CONTROL
C		Z-ADD*ZERO	W#SEQ 80	SEQ 计算记录条数
C		Z-ADD*ZERO	W#SEQ2 80	SEQ2 定位计算用
C**				
C		ENDSR		

\*\* #CMD  
PF03: 结束 PF12: 前画面 ENTER: 确认  
PF03: 结束 PF06: 打印 PF12: 前画面 ENTER: 确认



## 10.4 RPG 编程常用操作码详解

### 1 检验字符串“ CHECK （检验符）” “ CHECKR （反向检验）”

从右向左查找ST002第一个不等于空的字符的位数 @I=8

从左向右查找ST002第一个不等于'A'的字符的位数 @I=2

FMT	C	CLON01FACTOR1++++++OPCODE&EXTFACTOR2++++++RESULT++++++LEN++D+HI LOEQ
0000.18	C	MOVEL 'ABC' ST001 6
0000.19	C	MOVEL 'ABCDEFGH' ST002 8
0000.20	C	' ' CHECKR ST002 @I 5 0
0000.21	C	'A' CHECK ST002 @I 5 0

### 2 截取字符串

从ST002的第1位开始截取7位长的字段给ST003='ABCDEF'

从ST002的第2位开始截取4位长的字段给ST003='BCDE '

FMT	C	CLON01FACTOR1++++++OPCODE&EXTFACTOR2++++++RESULT++++++LEN++D+HI LOEQ
0000.24	C	7 SUBST ST002:1 ST003 6
0000.25	C	4 SUBST ST002:2 ST003 6

### 3. 字符项目转换

0000.28: 把数字型变量赋给字符型的变量空格部分会自动补零NUM002='00000909'

0000.30: 把'0'转换成' ' NUM003=' 9 9'

0000.31-0000.32: 转换中可能把数字中间的'0'变成' ', 所以从左边检索到第一个不能于空的字符的位数, 作为转换的起始位置, 把' '转换成'0'。

FMT	C	CLON01FACTOR1++++++OPCODE&EXTFACTOR2++++++RESULT++++++LEN++D+HI LOEQ
0000.27	C	Z-ADD 909 NUM001 8 0
0000.28	C	MOVE NUM001 NUM002 8
0000.30	C	'0':' ' XLATE NUM002 NUM003 8
0000.31	C	' ' CHECK NUM003 @I 5 0
0000.32	C	' ':'0' XLATE NUM003:@I NUM004 8

### 4. 连接字符串

将NAME1和LAST1连接成TEMP1 = 'MR. SMI ', LAST1:3 3代表两个字段间的空格数

默认为0, TEMP2 = 'MR.SMITH '

FMT	C	CLON01	FACTOR1++++++	OPCODE&EXT	FACTOR2++++++	RESULT++++++	LEN++	D+HI	LOEQ
0000.34	C			MOVE	'MR. '	NAME1	6		
0000.35	C			MOVE	'SMITH '	LAST1	6		
0000.36	C	NAME1		CAT	LAST1:3	TEMP1	9		
0000.37	C	NAME1		CAT	LAST1:0	TEMP2	9		

下面例子给出在因子 2 中的前置空格。在 CAT 之后，RESUL 中存 'MR. SMITH'  
如果这时因子2后面再跟参数就会在指定的空格数上加上因子2中的前置空格数。  
RESUL 中存 'MR. SMIT' 因子2的值就会被挤掉。

FMT	C	CLON01	FACTOR1++++++	OPCODE&EXT	FACTOR2++++++	RESULT++++++	LEN++	D+HI	LOEQ
0035.00	C			MOVE	'MR. '	NAME2	3		
0036.00	C			MOVE	' SMITH'	FIRST	6		
0037.00	C	NAME2		CAT	FIRST	RESUL	9		
0037.01	C	NAME2		CAT	FIRST:1	RESUL	9		

下面给出没有因子 1 的 CAT 操作，FLD2 有 9 个字符串，前面连接后  
它包括 'ABC '，FLD1 包括 'XYZ'，连接之后，FLD2 中有 'ABC XYZ'。

FMT	C	CLON01	FACTOR1++++++	OPCODE&EXT	FACTOR2++++++	RESULT++++++	LEN++	D+HI	LOEQ
0037.05	C			MOVE	'ABC'	FLD2	9		
0037.06	C			MOVE	'XYZ'	FLD1	3		
0037.07	C			CAT	FLD1:0	FLD2			

## 5. 扫描字符串

SCAN 在因子 2 中查找因子 1 起始位置 3 出现的串，因为 SCAN 操作是  
从位置 3 开始的，所以忽略基串位置 1 上的 Y。

操作结果把 5 和 6 放在数组第 1 和第 2 个元素中。指示器 90 为 ON

FMT	C	CLON01	FACTOR1++++++	OPCODE&EXT	FACTOR2++++++	RESULT++++++	LEN++	D+HI	LOEQ
0006.00	D	ARRAY	S		1 0 DIM(10)				
0057.00	C			MOVE	'YARRY'	FIELD1	6		
0058.00	C			MOVE	'Y'	FIELD2	1		
0059.00	C	FIELD2		SCAN	FIELD1:3	ARRAY			

SCAN 操作在因子 2 中从第 2 位开始查找，在因子 1 出现的 4 个长度的串，由于在 FIELD3  
中没找到 'T00L'，INT90 设为零，指示器 90 为 OFF。

在 FIELD5 中找到 'T00L'，INT91 设为4，指示器 91 为 ON。

FMT	C	CLON01FACTOR1++++++	OPCODE&EXTFACTOR2++++++	RESULT++++++	LEN++D+HI LOEQ
0054.00	C		MOVE ' TESTING'	FIELD3	7
0055.00	C		MOVE ' TEST00L'	FIELD5	7
0056.00	C		MOVEL ' T00L'	FIELD4	5
0057.00	C	FIELD4: 4	SCAN FIELD3: 2	INT90	2 0 90
0058.00	C	FIELD4: 4	SCAN FIELD5: 3	INT91	2 0 91

#### 6. 检查字段是否全是数字

如果字符串FIELD4全部为数字则\*IN21为'1' 否则为'0'

FMT	C	CLON01FACTOR1++++++	OPCODE&EXTFACTOR2++++++	RESULT++++++	LEN++D+HI LOEQ
0037.19	C		MOVE 12345	FIELD4	5
0037.20	C		TESTN	FIELD4	21
0037.21	C		MOVE 123	FIELD4	
0037.22	C		TESTN	FIELD4	21
0037.23	C		MOVE ' A1234'	FIELD4	
0037.24	C		TESTN	FIELD4	21

### 10.5 RPG 编程常用数组命令

#### 1. 数组排序（SORTA + 数组名）

FMT	C	CLON01FACTOR1++++++	OPCODE&EXTFACTOR2++++++	RESULT++++++	LEN++D+HI LOEQ
0007.00	D	#AR2	S	5 DIM(10)	
0008.00	D	#AR3	S	5 0 DIM(10)	
0075.00	C		CLEAR	#AR2	
0076.00	C		CLEAR	#AR3	
0077.00	C		MOVEL ' A0001'	#AR2(1)	
0078.00	C		MOVEL ' A0003'	#AR2(2)	
0079.00	C		MOVEL ' A0005'	#AR2(3)	
0080.00	C		MOVEL ' A0007'	#AR2(4)	
0081.00	C		MOVEL ' A0009'	#AR2(5)	
0082.00	C		MOVEL ' A0002'	#AR2(6)	
0083.00	C		MOVEL ' A0004'	#AR2(7)	
0084.00	C		MOVEL ' A0006'	#AR2(8)	
0085.00	C		MOVEL ' A0008'	#AR2(9)	
0086.00	C		MOVEL ' A0000'	#AR2(10)	
0087.00	C	*数组排序			
0088.00	C		SORTA	#AR2	

排序前后的效果

```
Evaluate Expression

Previous debug expressions
> EVAL #AR2
  #AR2(1) = 'A0001'
  #AR2(2) = 'A0003'
  #AR2(3) = 'A0005'
  #AR2(4) = 'A0007'
  #AR2(5) = 'A0009'
  #AR2(6) = 'A0002'
  #AR2(7) = 'A0004'
  #AR2(8) = 'A0006'
  #AR2(9) = 'A0008'
  #AR2(10) = 'A0000'

Debug . . .
F3=Exit   F9=Retrieve   F12=Cancel   F16=Repeat find   F19=Left   F20=Right
F21=Command entry   F23=Display output

Bottom
```

排序后的效果

```
Evaluate Expression

Previous debug expressions

  #AR2(8) = 'A0006'
  #AR2(9) = 'A0008'
  #AR2(10) = 'A0000'
> EVAL #AR2
  #AR2(1) = 'A0000'
  #AR2(2) = 'A0001'
  #AR2(3) = 'A0002'
  #AR2(4) = 'A0003'
  #AR2(5) = 'A0004'
  #AR2(6) = 'A0005'
  #AR2(7) = 'A0006'
  #AR2(8) = 'A0007'
  #AR2(9) = 'A0008'
  #AR2(10) = 'A0009'

Debug . . .

F3=Exit   F9=Retrieve   F12=Cancel   F16=Repeat find   F19=Left   F20=Right
F21=Command entry   F23=Display output
Already at bottom of area.

Bottom
```

## 2. 数组求和 XF00T+数组名+合计结果字段

FMT	C	CLON01FACTOR1++++++	OPCODE&EXTFACTOR2++++++	RESULT++++++	LEN++D+HI LOEQ
0007.00	D	#AR2	S	5	DIM(10)
0008.00	D	#AR3	S	5	0 DIM(10)
0075.00	C		CLEAR		#AR2
0076.00	C		CLEAR		#AR3
0090.00	C		Z-ADD	1	#AR3(1)
0091.00	C		Z-ADD	3	#AR3(2)
0092.00	C		Z-ADD	5	#AR3(3)
0093.00	C		Z-ADD	7	#AR3(4)
0094.00	C		Z-ADD	9	#AR3(5)
0095.00	C		Z-ADD	2	#AR3(6)
0096.00	C		Z-ADD	4	#AR3(7)
0097.00	C		Z-ADD	6	#AR3(8)
0098.00	C		Z-ADD	8	#AR3(9)
0099.00	C		Z-ADD	10	#AR3(10)
0100.00	C	*数组求和	1+3+5+7+9+2+4+6+8+10=55		
0102.00	C		XFOOT	#AR3	W@001 10 0

### Evaluate Expression

Previous debug expressions

```
#AR3(10) = 00010.
> EVAL W@001
W@001 = 0000000055.
> EVAL #AR3
#AR3(1) = 00001.
#AR3(2) = 00003.
#AR3(3) = 00005.
#AR3(4) = 00007.
#AR3(5) = 00009.
#AR3(6) = 00002.
#AR3(7) = 00004.
#AR3(8) = 00006.
#AR3(9) = 00008.
#AR3(10) = 00010.
```

Bottom

Debug . . .

F3=Exit F9=Retrieve F12=Cancel F16=Repeat find F19=Left F20=Right  
F21=Command entry F23=Display output

## 10.6 Query/400 创建高级查询

在“ AS/400 主菜单” 的命令行上，按以下屏幕所示输入 DSP0BJD 命令，然后按“ 执行” 键。这将创建名为 QRYFILE 的文件并将它放入 QGPL 库中。

主菜单	AS/400 主菜单	系统: RCH38342
选择下列其中一项:		
<ul style="list-style-type: none"><li>1. 用户任务</li><li>2. 办公室任务</li><li>3. 一般系统任务</li><li>4. 文件、库和文件夹</li><li>5. 程序设计</li><li>6. 通信</li><li>7. 定义或更改系统</li><li>8. 问题处理</li><li>9. 显示菜单</li> <li>90. 注销</li></ul>		
选择或命令		
==> DSP0BJD OBJ(QGPL/*ALL) OBJTYPE(*ALL ) OUTPUT(*OUTFILE) OUTFILE(QGPL/QRYFILE)		
F3=退出 F4=提示 F9=检索 F12=取消 F13=用户支持		
F23=设置初始菜单		

在命令行上输入 WRKQRY，然后按“ 执行” 键。

主菜单	AS/400 主菜单	系统: RCH38342
选择下列其中一项:		
<ul style="list-style-type: none"><li>1. 用户任务</li><li>2. 办公室任务</li><li>3. 一般系统任务</li><li>4. 文件、库和文件夹</li><li>5. 程序设计</li><li>6. 通信</li><li>7. 定义或更改系统</li><li>8. 问题处理</li><li>9. 显示菜单</li> <li>90. 注销</li></ul>		
选择或命令		
==> WRKQRY		
F3=退出 F4=提示 F9=检索 F12=取消 F13=用户支持		
F23=设置初始菜单		

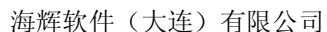


显示“使用查询”屏幕。在该屏幕上选择选项 1（创建）。然后在查询提示中输入一个名称，并在库提示中输入一个名称，指定存储您的查询的库。按“执行”键。

使用查询			
输入选项，按“执行”键。			
选项	1	1=创建，2=更改，3=复制，4=删除 5=显示，6=打印定义 8=批处理运行 9=运行	
查询	QNAME	名称、按 F4 获得列表	
库	YOURLIB	名称、*LIBL、按 F4 获得列表	
F3=退出      F4=提示      F5=刷新      F12=取消			

在“定义查询”屏幕上，通过在每个定义步骤左面输入 1，选择定义步骤：选择并排序字段、选择记录、选择排序字段、指定报告列格式、选择报告汇总函数和定义报告细分。（注意指定文件选择已经为您选定。）按“执行”键。

定义查询			
查询	QNAME	选项	CREATE
库	YOURLIB	CCSID	37
输入选项，按“执行”键。按 F21 选择全部。			
1=选择			
Opt	查询定义选项		
1	指定文件选择		
	定义结果字段		
1	选择并排序字段		
1	选择记录		
1	选择排序字段		
	选择整理顺序		
1	指定报告列格式		
1	选择报告汇总函数		
1	定义报告细分		
	选择输出类型和输出格式		
	指定处理选项		
F3=退出      F5=报告      F12=取消			
F13=布局      F18=文件      F21=全选			



在“指定文件选择”屏幕即“定义查询”屏幕后显示的第一个屏幕上，在文件提示中输入 QRYFILE，并在库提示中输入 QGPL。成员和格式提示已经填好。按“执行”键。

输入选项，按“ 执行” 键。按 F9 指定附加的文件选择。

F3=退出      F4=提示      F5=报告      F9=添加文件  
F12=取消      F13=布局      F24=其他键

出现信息“选择文件，或按“执行”键确认。”。

输入选项，按“ 执行” 键。按 F9 可添加。

F3=退出                  F4=提示                  F5=报告                  F9=添加文件  
F12=取消                F13=布局                F24=其他键

选择文件，或按“ 执行” 键确认。



再次按“ 执行” 键。

下一个显示的屏幕是“ 选择并排序字段” 屏幕。

选择并排序字段					
为要显示于报告中的最多 500 个字段的名称输入序号 (0-9999), 然后按“ 执行” 键。					
序号	字段	序号	字段	序号	字段
	ODDCEN		ODCCEN		ODSV02
	ODDDAT		ODCDAT		ODSV03
	ODDTIM		ODCTIM		ODSV04
	ODLBNM		ODOBOW		ODSV05
	ODOBNM		ODSCEN		ODSV06
	ODOBTP		ODSDAT		ODSV07
	ODOBAT		ODSTIM		ODSV08
	ODOBFR		ODSCMD		ODSV09
	ODOBSZ		ODSSZE		ODSV10
	ODOBTX		ODSSLT		ODSVMR
	ODOBLK		ODSDEV		ODRCEN
	ODOBDM		ODSV01		ODRDAT
尚有...					
F3=退出		F5=报告		F11=显示文本	
F13=布局		F20=重新编号		F12=取消	
				F24=其他键	

如果您的屏幕是多列格式（即，如果文本、Len 和 Dec 列不显示），则按 F11（显示文本）来显示有关列表中字段的某些其他信息。在每个字段左面输入数字 1 至 5，选择字段 ODLBNM、ODOBTP、ODOBAT、ODOBSZ 和 ODOBTX，如下所示。您选择的字段将按照指定的次序在查询报告中出现（ODLBNM 首先出现，接着是 ODOBTP，依此类推）。

选择并排序字段					
为要显示于报告中的最多 500 个字段的名称输入序号 (0-9999), 然后按“ 执行” 键。					
序号	字段	文本	Len	Dec	
	ODDCEN	显示世纪	1		
	ODDDAT	显示日期: 格式- MMDDYY	6	x	
	ODDTIM	显示时间	6		
1	ODLBNM	库	10		
	ODOBNM	对象	10		
2	ODOBTP	对象类型	8		
3	ODOBAT	对象属性	10		
	ODOBFR	存储器释放: 0-未释放, 1-已释放	1		
4	ODOBSZ	对象大小	10	0	
5	ODOBTX	文本说明	50		
	ODOBLK	对象锁定: 0-未锁定, 1-已锁定	1		
	ODOBDM	对象损坏: 0-未损坏, 1-损坏	1		
尚有...					
F3=退出		F5=报告		F11=仅显示名称	
F13=布局		F20=重新编号		F12=取消	
				F24=其他键	

按“ 执行” 键。查询重新安排屏幕上的字段，以便您选择的字段按您指定的次序显示在列表首。  
还显示“ 按“ 执行” 键确认。” 信息。

选择并排序字段				
为要显示于报告中的最多 500 个字段的名称输入序号 (0-9999)， 按“ 执行” 键。				
序号	字段	文本	Len	Dec
1	ODLBNM	库	10	
2	ODOBTP	对象类型	8	
3	ODOBAT	对象属性	10	
4	ODOBSZ	对象大小	10	0
5	ODOBTX	文本说明	50	
	ODDCEN	显示世纪	1	
	ODDDAT	显示日期: 格式- MMDDYY	6	
	ODDTIM	显示时间	6	
	ODOBNM	对象	10	
	ODOBFR	存储器释放: 0-未释放, 1-已释放	1	
	ODOBLK	对象锁定: 0-未锁定, 1-已锁定	1	
	ODOBDM	对象损坏: 0-未损坏, 1-损坏	1	
			尚有...	
F3=退出		F5=报告	F11=仅显示名称	F12=取消
F13=布局		F20=重新编号	F21=全选	F24=其他键

再次按“ 执行” 键。

下一个显示的屏幕是“ 选择记录” 屏幕。在此处指定希望在报告中包括哪些记录。

选择记录				
输入比较, 按“ 执行” 键。指定 OR 以启动每个新组。				
测试: EQ, NE, LE, GE, LT, GT, RANGE, LIST, LIKE, IS, ISNOT...				
AND/OR	字段	测试	值 (字段、数字或 '字符')	
底部				
字段	文本	Len	Dec	
ODLBNM	库	10		
ODOBTP	对象类型	8		
ODOBAT	对象属性	10		
ODOBSZ	对象大小	10	0	
ODOBTX	文本说明	50		
			尚有...	
F3=退出		F5=报告	F9=插入	F11=仅显示名称
F12=取消		F13=布局	F20=重组	F24=其他键



您希望在报告中包括具有 \*FILE 或 \*PGM 对象类型的对象的所有记录。在字段、测试和值列中输入信息，如下屏幕所示。暂时不要按“执行”键。

选择记录			
输入比较，按“执行”键。指定 OR 以启动每个新组。			
测试： EQ, NE, LE, GE, LT, GT, RANGE, LIST, LIKE, IS, ISNOT...			
AND/OR	字段	测试	值（字段、数字或 ' 字符'）
	ODOBTP	LIST	'*FILE' '*PGM'
			底部
字段	文本	Len	Dec
ODLBNM	库	10	
ODOBTP	对象类型	8	
ODOBAT	对象属性	10	
ODOBSZ	对象大小	10	0
ODOBTX	文本说明	50	
			尚有...
F3=退出	F5=报告	F9=插入	F11=仅显示名称
F12=取消	F13=布局	F20=重组	F24=其他键

现在按 F5（报告）显示报告。“显示报告”屏幕出现，显示您的查询报告，它基于您到目前为止所定义的查询。（在屏幕上看到的信息取决于当前在系统上的 QGPL 中的对象。您看到的可能与以下屏幕中显示的不尽相同。）

显示报告					
报告宽度 . . . . .					100
定位到行 . . . . .	移位到列 . . . . .				
行 . . . + . . . 1 . . . + . . . 2 . . . + . . . 3 . . . + . . . 4 . . . + . . . 5 . . . + . . . 6 . . . + . . . 7 . .					
库	对象类型	对象属性	对象大小	文本说明	
000001 QGPL	*PGM	CLP	14,336	B & R 示例 - 第 2 页	
000002 QGPL	*PGM	CLP	16,384	B & R 实例 - 第 2 页	
000003 QGPL	*FILE	PF	8,192		
000004 QGPL	*FILE	PF	1,024	缺省源数据	
000005 QGPL	*FILE	PF	1,024	缺省源数据	
000006 QGPL	*FILE	PF	16,384	缺省源数据	
000007 QGPL	*FILE	DKTF	2,560	缺省软盘数据	
000008 QGPL	*FILE	DKTF	2,560	缺省源软盘	
000009 QGPL	*FILE	PF	1,024	缺省源数据	
000010 QGPL	*FILE	PF	140,288	RSTS36FLR 命令	
000011 QGPL	*FILE	PRTF	2,048	缺省假脱机输出	
000012 QGPL	*FILE	PRTF	2,048	缺省假脱机打印	
000013 QGPL	*FILE	PRTF	2,048	缺省假脱机打印	
000014 QGPL	*FILE	PF	38,912	DSPOBJD 的输出文件	
000015 QGPL	*FILE	PF	16,384		
			尚有...		
F3=退出	F12=取消	F19=向左	F20=向右	F21=分割	F22=宽度 80

在屏幕底部的右侧，出现尚有... 信息。这表示报告内容无法全部显示在屏幕上，因此如果您希望查看报告全部内容，可使用翻页键或 F20（向右）和 F19（向左）来浏览报告（既可以从左向右，也可以从上到下）。当查看完报告后，按 F3（退出）返回到“选择记录”屏幕。

选择记录			
输入比较，按“ 执行” 键。指定 OR 以启动每个新组。			
测试： EQ, NE, LE, GE, LT, GT, RANGE, LIST, LIKE, IS, ISNOT...			
AND/OR	字段	测试	值（字段、数字或 ' 字符'）
	ODOBTP	LIST	' *FILE' ' *PGM'
			底部
字段	文本	Len	Dec
ODLBNM	库	10	
ODOBTP	对象类型	8	
ODOBAT	对象属性	10	
ODOBSZ	对象大小	10	0
ODOBTX	文本说明	50	
			尚有...
F3=退出	F5=报告	F9=插入	F11=仅显示名称
F12=取消	F13=布局	F20=重组	F24=其他键

在“选择记录”屏幕上按“ 执行” 键。

下一个显示的屏幕是“选择排序字段”屏幕。在此处指定希望 Query 使用哪些字段来将为报告选择的记录排序。如果希望首先按对象类型，然后按对象大小对记录排序，则在 ODOBTP 旁输入 1，在 ODOBSZ 旁输入 2，如下所示。

选择排序字段			
为多达 32 个字段的名称输入排序优先级 (0-999) 以及 A（升序）或 D（降序），然后按“ 执行” 键。			
排序			
优先	A/D	字段	文本
		ODLBNM	库
1		ODOBTP	对象类型
		ODOBAT	对象属性
2		ODOBSZ	对象大小
		ODOBTX	文本说明
			底部
F3=退出	F5=报告	F11=仅显示名称	F12=取消
F13=布局	F18=文件	F20=重新编号	F24=其他键

按“ 执行” 键。查询重新安排屏幕上的字段，以便为排序选择的字段按您指定的次序显示在列表首。还显示“ 按“ 执行” 键确认。” 信息。

选择排序字段					
为多达 32 个字段的名称输入排序优先级 (0-999) 以及 A (升序) 或 D (降序)，然后按“ 执行” 键。					
排序	优先	A/D	字段	文本	Len Dec
1	A		ODOBTP	对象类型	8
2	A		ODOBSZ	对象大小	10 0
			ODLBNM	库	10
			ODOBAT	对象属性	10
			ODOBTX	文本说明	50
底部					
F3=退出		F5=报告		F11=仅显示名称	
F13=布局		F18=文件		F20=重新编号	
				F12=取消	
				F24=其他键	

再次按“ 执行” 键。

下一个屏幕是“ 指定报告列格式” 屏幕。在此处指定列间距、列标题以及其他报告格式选项。

指定报告列格式					
输入信息，按“ 执行” 键。					
列标题: *NONE、对齐的文本行					
字段	列间距	列标题	Len	Dec	编辑
ODLBNM	0	库	10		
ODOBTP	2	对象类型	8		
ODOBAT	2	对象属性	10		
尚有...					
F3=退出		F5=报告		F10=处理 / 前一屏	
F13=布局		F16=编辑		F18=文件	
				F12=取消	
				F23=长注释	



Query 为您提供很多这类信息。更改 ODLBNM 字段的列标题，使它显示为库名而不是库。

指定报告列格式					
输入信息，按“ 执行” 键。					
列标题： *NONE、对齐的文本行					
字段	列 间距	列标题	Len	Dec	编辑
ODLBNM	0	库名	10		
ODOBTP	2	对象 类型	8		
ODOBAT	2	对象 属性	10		
					尚有...
F3=退出 F13=布局	F5=报告 F16=编辑	F10=处理 / 前一屏 F18=文件	F12=取消 F23=长注释		

按“ 执行” 键。

您看到的下一个屏幕是“ 选择报告汇总函数” 。通过在此屏幕上指定选项，可在报告中对选择的字段汇总--即对于选择的字段，您可指定在报告中包括总计、平均值、最小值、最大值以及（或）计数。

选择报告汇总函数					
输入选项，按“ 执行” 键。					
1=总计    2=平均值    3=最小值    4=最大值    5=计数					
---选项-----	字段	文本	Len	Dec	
- - - - -	ODLBNM	库	10		
- - - - -	ODOBTP	对象类型	8		
- - - - -	ODOBAT	对象属性	10		
- - - - -	ODOBSZ	对象大小	10	0	
- - - - -	ODOBTX	文本说明	50		
					底部
F3=退出 F12=取消	F5=报告 F13=布局	F10=处理 / 前一屏 F18=文件	F11=仅显示文本 F23=长注释		



通过在字段旁输入 1 至 5，指定您希望对字段 ODOB SZ 应用所有汇总函数（总计、平均值、最小值、最大值和计数），如以下屏幕所示。

选择报告汇总函数									
输入选项，按“ 执行” 键。									
1=总计    2=平均值    3=最小值    4=最大值    5=计数									
---选项---		字段	文本					Len	Dec
		ODLBNM	库					10	
		ODOBTP	对象类型					8	
		ODOBAT	对象属性					10	
1	2	3	4	5	ODOBSZ	对象大小		10	0
		ODOBTX	文本说明					50	
底部									
F3=退出		F5=报告		F10=处理 / 前一屏		F11=仅显示文本			
F12=取消		F13=布局		F18=文件		F23=长注释			

按“ 执行” 键。

下一个要显示的屏幕是“ 定义报告细分” 屏幕。在该屏幕上指定希望使用哪些字段作为细分字段。报告细分用于在每次报告细分字段的值更改时将报告划分为几个记录组。

定义报告细分									
为多达 9 个字段名输入细分级（1-6），然后按“ 执行” 键。									
（对每个细分级使用需要的任意多字段。）									
细分级	排序	字段	文本					Len	Dec
	优先	ODLBNM	库					10	
	10	ODOBTP	对象类型					8	
		ODOBAT	对象属性					10	
	20	ODOBSZ	对象大小					10	0
		ODOBTX	文本说明					50	
底部									
F3=退出		F5=报告		F10=处理 / 前一屏		F11=仅显示名称			
F12=取消		F13=布局		F18=文件		F23=长注释			

在左边的细分级列中，为字段 ODOBTP 输入 1，以指定细分级 1。

定义报告细分					
为多达 9 个字段名输入细分级 (1-6)，按“ 执行” 键。 (对每个细分级使用需要的任意多字段。)					
细分级	排序 优先	字段	文本	Len	Dec
1	10	ODLBNM	库	10	
		ODOBTP	对象类型	8	
		ODOBAT	对象属性	10	
	20	ODOBSZ	对象大小	10	0
		ODOBTX	文本说明	50	
底部					
F3=退出		F5=报告		F10=处理 / 前一屏	
F12=取消		F13=布局		F11=仅显示名称	
		F18=文件		F23=长注释	

按“ 执行” 键。

在下一个屏幕“ 格式化报告细分” 上，为您定义的报告细分指定期望的格式。注意细分级提示中的值为零。可使用细分级 0 将所有指定的汇总函数的最终汇总值打印在报告的结尾。对于此示例，不要在此屏幕上做任何更改。只须按“ 执行” 键。

格式化报告细分	
细分级	0
输入选项，按“ 执行” 键。 (在文本中输入 &field 以插入细分值。)	
抑制汇总	N Y=是，N=否
细分文本	最终总计
级别	字段
1	ODOBTP
F3=退出	F5=报告
F13=布局	F18=文件
F10=处理 / 前一屏	F12=取消
F23=长注释	





在下一个屏幕上，可看到细分级提示已添上 1。在此处对细分级 1 格式化报告细分。在细分文本提示中，输入对象类型的细分文本。该文本将在每次发生此细分级的报告细分时出现。

格式化报告细分			
细分级别	. . . . . : 1		
输入选项，按“ 执行” 键。 (在文本中输入 &field 以插入细分值。)			
跳至新页	. . . . . N	Y=是, N=否	
抑制汇总	. . . . . N	Y=是, N=否	
细分文本	. . . . . 对象类型的细分文本		
级别	字段		
1	ODOBTP		
F3=退出	F5=报告	F10=处理 / 前一屏	F12=取消
F13=布局	F18=文件	F23=长注释	

按“ 执行” 键。

已完成您先前选择的所有定义步骤，因此“ 定义查询” 屏幕再次出现。(您先前选择的定义步骤现在用左侧的 > 符号表示。)

定义查询			
查询	. . . . . : QNAME	选项	. . . . . : CREATE
库	. . . . . : YOURLIB	CCSID	. . . . . : 37
输入选项，按“ 执行” 键。按 F21 选择全部。 1=选择			
Opt	查询定义选项 > 指定文件选择 定义结果字段 > 选择并排序字段 > 选择记录 > 选择排序字段 选择整理顺序 > 指定报告列格式 > 选择报告汇总函数 > 定义报告细分 选择输出类型和输出格式 指定处理选项		
F3=退出	F5=报告	F12=取消	
F13=布局	F18=文件	F21=全选	



现在按 F5（报告）显示报告。“显示报告”屏幕出现，显示您完成的查询报告。（在屏幕上看到的信息取决于当前在系统上的 QGPL 中的对象。您看到的可能与以下屏幕中显示的不尽相同。）

显示报告						
				报告宽度 . . . . .	:	104
定位到行 . . . . .				移位到列 . . . . .		
行 . . . . .	1 . . . . .	2 . . . . .	3 . . . . .	4 . . . . .	5 . . . . .	6 . . . . .
对象	对象	对象	对象	对象	文本说明	
名	类型	属性		大小		
000001 QGPL	*FILE	PF		1,024	缺省源数据	
000002 QGPL	*FILE	PF		1,024	缺省源数据	
000003 QGPL	*FILE	PF		1,024	缺省源数据	
000004 QGPL	*FILE	PF		1,024	缺省源数据	
000005 QGPL	*FILE	PF		1,024		
000006 QGPL	*FILE	DSPF		1,536		
000007 QGPL	*FILE	PRTF		2,048	缺省假脱机输出	
000008 QGPL	*FILE	PRTF		2,048	缺省假脱机打印	
000009 QGPL	*FILE	PRTF		2,048	缺省假脱机打印	
000010 QGPL	*FILE	TAPF		2,048	缺省磁带数据	
000011 QGPL	*FILE	TAPF		2,048	缺省源磁带	
000012 QGPL	*FILE	DKTF		2,560	缺省软盘数据	
000013 QGPL	*FILE	DKTF		2,560	缺省源软盘	
000014 QGPL	*FILE	PF		8,192		
000015 QGPL	*FILE	PF		8,192		
						尚有...
F3=退出	F12=取消	F19=向左	F20=向右	F21=分割	F22=宽度 80	

在屏幕底部的右侧，出现尚有... 信息。这意味着报告的全部内容无法显示在屏幕上。使用翻页键或 F20（向右）和 F19（向左）浏览报告（从左向右和从上向下均可），以查看报告中的细分和汇总。当查看完报告时，按 F3（退出）返回到“定义查询”屏幕。

定义查询			
查询 . . . . .	QNAME	选项 . . . . .	CREATE
库 . . . . .	QGPL	CCSID . . . . .	37
输入选项，按“执行”键。按 F21 选择全部。			
1=选择			
Opt	查询定义选项		
>	指定文件选择		
	定义结果字段		
>	选择并排序字段		
>	选择记录		
>	选择排序字段		
	选择整理顺序		
>	指定报告列格式		
>	选择报告汇总函数		
>	定义报告细分		
	选择输出类型和输出格式		
	指定处理选项		
F3=退出	F5=报告	F12=取消	
F13=布局	F18=文件	F21=全选	

按 F3（退出）。

显示“退出此查询”屏幕。如果不希望保存查询或再次运行它，在此屏幕上将保存定义提示更改为 N（否），将运行选项提示更改为 3（不要运行）。

退出此查询			
输入选项，按“执行”键。			
保存定义 . . . . .	N	Y=是，N=否	
运行选项 . . . . .	3	1=交互式运行 2=批处理运行 3=不运行	
对于保存的定义：			
查询 . . . . .	QNAME	名称	
库 . . . . .	QGPL	名称、按 F4 获得列表	
文本 . . . . .			
权限 . . . . .	*CHANGE	*LIBCRTAUT、*CHANGE、*ALL *EXCLUDE、*USE 授权表名	
F4=提示	F5=报告	F13=布局	F14=定义查询

按“执行”键。

“使用查询”屏幕出现，显示信息“查询选项处理成功完成”。

使用查询			
输入选项，按“执行”键。			
选项 . . . . .		1=创建，2=更改，3=复制，4=删除 5=显示，6=打印定义 8=批处理运行 9=运行	
查询 . . . . .	QNAME	名称、按 F4 获得列表	
库 . . . . .	QGPL	名称、*LIBL、按 F4 获得列表	
F3=退出	F4=提示	F5=刷新	F12=取消

现在可按 F3（退出）结束此示例，返回到“AS/400 主菜单”。