

# C#言語による在庫管理システムの開発\*

外山 真也\*<sup>1</sup>・柳田 卓男\*<sup>2</sup>・清島 由美\*<sup>2</sup>

## Development of Inventory Management System by C#

Masaya TOYAMA, Takuo YANAGITA and Yumi KIYOSHIMA

株式会社正洋は、BASIC 言語で開発した在庫管理システムを運用していた。しかし、データの増大に伴い、検索や集計に時間を要するようになったこと、業務改善による機能変更などの対応が困難になったことなどがあり、C#言語による開発を検討することとなった。

今回、工業技術センターが開発してきている各種のライブラリを利用することで短期間にシステムの構築を実現できたので紹介する。

キーワード：データベース、在庫管理、ソフトウェア、C#

### 1 はじめに

株式会社正洋は、平成 10 年頃から BASIC 言語により開発した在庫管理システムを利用していたが、その後のデータの増大に伴い、検索や集計に時間を要するようになったこと、業務内容の変化などによる機能改善などの要望に対応することが困難になったこと等により、MZPF を利用したシステムの開発を検討した。

しかし、MZPF を利用したシステムの場合、帳票出力などの機能において困難が予想されたため、急遽 C#言語による開発を検討することとなった。

今回、工業技術センターが開発してきている各種のライブラリを利用することで、短期間にシステムの構築を実現できた。

### 2 開発方法

開発したシステムの初期起動状況を図 1 に示す。開発は、主に 1) 依頼データ処理、2) 納品データ処理、3) 受払いデータ処理、4) 在庫管理処理の四つの機能に分けられる。以下において各開発手順を示す。

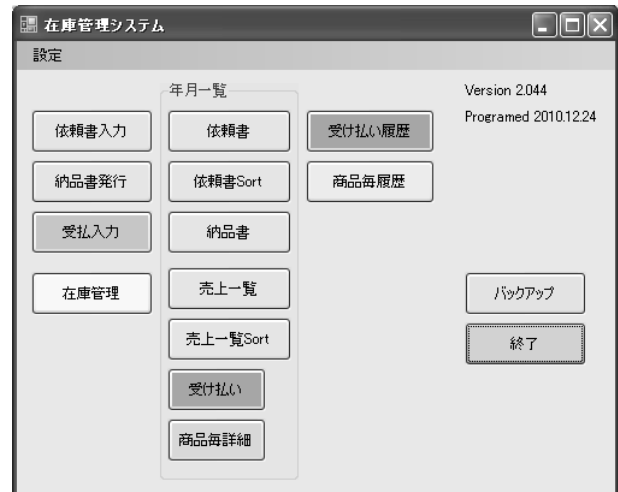


図 1 初期起動状況



図 2 依頼データ処理

#### 2-1 依頼データ処理

依頼データ入力は、発注企業からの注文データを入力する機能部分である。そのデータ入力

\* 共同研究 MZPF を利用した生産管理システムの構築

\*1 機械電子部

\*2 株式会社 正洋

状態を図2に示す。このとき、下部にデータ一覧を示し、入力する項目を一覧から選択することも可能にした。また、受注数を入力した後、この依頼データに関する納品が発生した場合には、注文数から納品数を差し引いた数を「残数」に、その伝票リストの番号などを「伝票リスト」に表示するようにした。

このことにより、依頼データ入力において、その依頼書のデータにおける納品しなければならない残数が、わかるようになった。

### 2-2 納品データ処理

納品データ処理の状態を図3に示す。この処理では、まず依頼番号を入力する。このとき、依頼番号に登録された受注データが下部の一覧に表示される。この一覧に表示されたデータを選択して、その部品が納品される数値などを入力できるようにした。

ここで入力されたデータは、保存登録される際に、依頼番号で示された依頼データの伝票リストに追加登録されるようにした。このことにより、依頼データには、最新の残数が更新されるようにした。

また、この納品処理によって出力される納品伝票の印刷状態を図4に示す。この納品伝票は控えを含めて4種類の同様な書式を印刷するため、自動的に出力できるようにした。

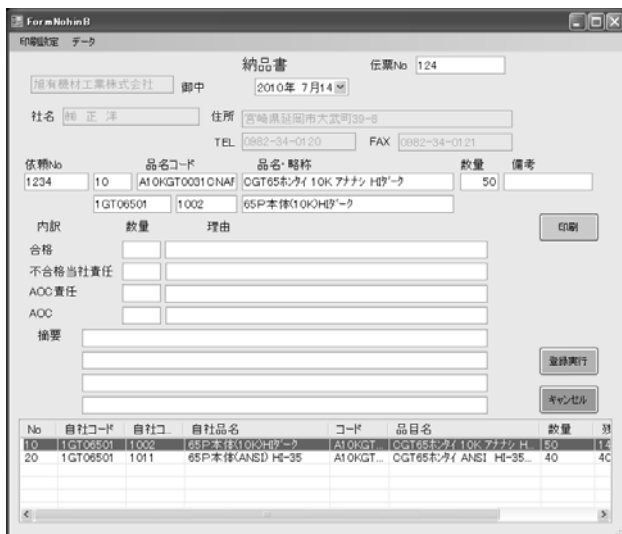


図3 納品データ処理

### 2-3 受払いデータ処理

作成した製品の納品だけでなく、発注企業からの受け入れなどが発生するため、受払い処理の機

能を追加した。この受払い処理の状態を図5に示す。

この処理において、下部にデータ一覧を表示し、その一覧からデータを選択して、受け入れ、払い出しの数値の入力編集を可能にした。



図4 納品伝票印刷処理



図5 受払いデータ処理



図6 在庫管理処理

### 2-4 在庫管理データ処理

在庫管理は、先月の在庫データ、納品データ、受払いデータを集計して、当月の在庫データが集

計される。在庫管理処理の起動状況を図6に示す。

在庫処理は、まず、マスターデータを当月の在庫データとして転写し、先月の在庫データを当月の繰越に転写する。次に、納品データ、受払いデータを当月の在庫データへ転写し、項目ごとに集計するようにした。

No.	品目No.	部品名	数量	繰越	受入	引出	千原A	千原B	千原C	在庫	
1	107005	101	05P-本体付(100)付	800	0	5	56	1	2	4	-57
2	107005	103	05P-本体付(100)付	1307	0	41	2	2	4	4	-50
3	107005	104	05P-本体付(100)付	1022	57	0	0	0	0	0	57
4	107005	105	05P-本体付(100)付	0	0	0	0	0	0	0	0
5	107005	106	05P-本体付(100)付	0	0	0	0	0	0	0	0
6	107005	107	05P-本体付(100)付	0	0	0	0	0	0	0	0
7	107005	108	05P-本体付(100)付	0	0	0	0	0	0	0	0
8	107005	109	05P-本体付(100)付	0	0	0	0	0	0	0	0
9	107005	110	05P-本体付(100)付	0	0	0	0	0	0	0	0
10	107005	111	05P-本体付(100)付	276	0	0	0	0	0	0	0
11	107005	112	05P-本体付(100)付	212	0	0	0	0	0	0	0
12	107005	113	05P-本体付(100)付	276	0	0	0	0	0	0	0
13	107005	121	05P-本体付(100)付	200	0	0	0	0	0	0	0
14	107005	124	05P-本体付(100)付	473	0	0	0	0	0	0	0
15	107005	133	05P-本体付(100)付	200	0	0	0	0	0	0	0
16	107005	134	05P-本体付(100)付	206	0	0	0	0	0	0	0
17	107005	135	05P-本体付(100)付	80	0	0	0	0	0	0	0
18	107005	136	05P-本体付(100)付	0	0	0	0	0	0	0	0
19	107005	137	05P-本体付(100)付	267	0	0	0	0	0	0	0
20	107005	138	05P-本体付(100)付	0	0	0	0	0	0	0	0
21	107005	139	05P-本体付(100)付	0	0	0	0	0	0	0	0
22	107005	140	05P-本体付(100)付	0	0	0	0	0	0	0	0
23	107005	141	05P-本体付(100)付	22	0	0	0	0	0	0	0
24	107005	142	05P-本体付(100)付	22	0	0	0	0	0	0	0
25	107005	143	05P-本体付(100)付	110	0	0	0	0	0	0	0
26	107005	145	05P-本体付(100)付	0	0	0	0	0	0	0	0
27	107005	146	05P-本体付(100)付	0	0	0	0	0	0	0	0
28	107005	148	05P-本体付(100)付	0	0	0	0	0	0	0	0
29	107005	201	05P-本体付(100)付	0	0	0	0	0	0	0	0
30	107005	202	05P-本体付(100)付	0	0	0	0	0	0	0	0

図7 在庫データの印刷処理

### 3 結果および考察

プログラム開発においては Microsoft Visual Studio.NET C# Express を利用した。各データのファイル名の書式は以下のようにした。

依頼データ = IRI 年月日-依頼番号

納品データ = DNP 年月日-伝票番号

受払データ = UKE 年月日

在庫データ = ZIK 年月

このようにデータファイル名称に年月日を入れることにより、データの検索を高速化することが

できた。また、依頼データにおける伝票リストにおいては、伝票データのファイル名を登録保存するようにして、データの軽減を図った。

### 4 まとめ

今回の開発において、各項目のデータは年月日と番号を基準にしたファイル名にして登録保存する手法をとった。

データベースを構築してデータを蓄積すると、データの増大に伴い検索に要する時間が長くなる欠点があった。

本手法では、データがファイルごとに分割されているため、検索処理に関して大きな影響を受けないこと、また不要なデータを別のフォルダに移動させることにより、処理の軽減が実現され、維持管理も容易になることがわかった。

今回、開発に利用したソフトは主に Microsoft Visual Studio.NET C# Express で、無料のソフトである。そしてオブジェクト指向による開発を実施し、操作性の良いシステムが構築できたと考える。

現在、このシステムの運用を始めているが、この3ヶ月は順調に稼働している。