装置制御へのZIPC適用について

株式会社アドテックエンジニアリング
技術本部　システム技術部
課長　橋本　一範

はじめに

近年、産業用生産装置の業界においても、高機能で品質の良いものを短期間で納入することが重要になってきている。

このため、設計/コーディング時間の短縮だけではなく、品質の向上も重点課題となっている。

そもそも、工場で使用する生産機器は、お客様毎の特別仕様が多く、そのためソフトウェアも複雑になりやすい。どんなに複雑なソフトウェアを作成してもバグが無ければ結構であるが、実際にそのようなことは無く、複雑になればなるほどバグの数は急激に増えてゆくことはもはや常識である。

無くせないバグと向き合うためには、バグが表に出にくい設計を行うことと、致命的なバグは予め出し尽くすことが大切で、これは設計と出荷前にどれだけテストを行ったかにかかってくる。

このテストとは出来上がった完成品の動作テストだけではなく、設計時点で設計に潜むバグを見つけ出すことも含んでいる。基本的には前工程であればあるほど、バグの検出・修正は容易になる。

これらのことを前提として、我々は自社製品の制御ソフトの開発にZIPCの適用を試みた。

装置の構成

話しを分かりやすくするために、我々の主力製品であるプリント基板用自動露光装置の機械構成と制御構成を簡単に説明する。

装置は大きく分けて

1. 搬入部　前工程からの基板の受取
2. 露光室　マスクとの位置合わせと露光
3. 搬出部　後工程への基板引渡し
4. 搬送部　搬入－露光室－搬出間の基板搬送

の４部構成になっている（図１）。

図１

前工程から材料が搬送されてくるサイクルタイムは任意でありばらつきがある。また、後工程が製品を受け取れるサイクルタイムもばらつきを持っている。

このため、搬入部は前工程の、搬出部は後工程の、露光室は自分自身の工程のタイミングに依存しており、搬送部が各部の間を取り持っている構造となっている。

次に制御構成だが、この装置には制御用/HMI用/画像処理用の３つのコンピューターを使用している（図２）。

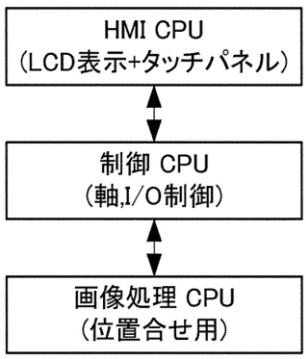


図２

今回ZIPCを適用するのは制御用のコンピュータプログラムである。

装置にはサーボ/パルスモーターが16軸程度、I/O点数が200点程度装備されている。

プログラミング言語はGCCを使用する。

適用にあたり

適用にあたり検討をしたところ、以下のような問題点、指摘が各担当から上がった。

・装置制御にツールは向かないのでは？
・ツールは正常に動くのか？
・もし原因不明のバグが出たときにツールの異常かどうか判断できるのか？
・ソースコードが大きくなるのでは？

いずれの問題も、今まで使ったことのないツールというものに対しての、不安感や嫌悪感が源となっており、特にベテランのエンジニアほどこの傾向が強い。そこで初回は若手のエンジニアを中心にプロジェクトを組み、ZIPCが「使えるか/使えないか」ではなく、「どうすれば使えるか」ということを考えながら、我々に合った適用方法を考えることとした。

その結果、ZIPCをコードの全てに適用するのは適当でないという結論に至った。

複数の状態を持ち、状態間を行き来するような部分には適用価値があるのだが、極めてシーケンシャルな動作をする部分については、余り意味がなく、デバッグしづらくなるだけであると考えた。

例えば、上位関数から呼び出され、エアシリンダ1,2,3を順番に制御する場合、バルブ1をONしてから一定時間以内にセンサー1がONすればバルブ2以降の制御へ、OFFのままなら中断してエラーを上位に返すといった関数になる。

この場合、フローチャートで十分シンプルに記述可能であり、実機にデバッガを接続し、動作を確認する際にもフローのほうが分かりやすい（図3）。

また、I/Oやサーボ軸などの制御ドライバーは、省配線化メーカーに委託するため、ここも適用から除外した。

次に適用しやすくするためのソフト構造を考えた。まずイベントドリブンのSTMが最も分かりやすく、適していると思われたので、タスク間やユニット間の通信は、全てOS（iTRON）のメールとし、このメール受信をイベントとして動作する構造とした。

また、フローで設計したほうがいいと思われる、シーケンシャルな動作を「個別動作関数」として、原則「正常時は0、異常時はエラーコード」を返す関数群としてまとめることとした。

タスクは機械構造に合せて分け、それぞれが独立して動けるようにした（図4）。

また、I/Oや軸が省配線化してあり、全てを通信で制御するため、メーカーに依頼し、この通信をエミュレートし実際に装置を制御しているかのごとく動作する「省配線I/Oシミュレーター」を作成した。

作成にあたり

従来、このような装置を開発する場合は、ユニットごとに縦割りで設計することが多かったが、今回は横割りで担当の割り振りを行った。

つまり、ZIPC担当、個別動作担当、ドライバー担当と割り振った。

また、外部に委託可能な部分は出来るだけ委託することとした。

ZIPCではシミュレーション、ジェネレーションともに、きちんと使用することとした。

個別動作関数はフローチャートを作成した後、エディタを使ってコーディングし、ROM-ICEと省配線シミュレーターを

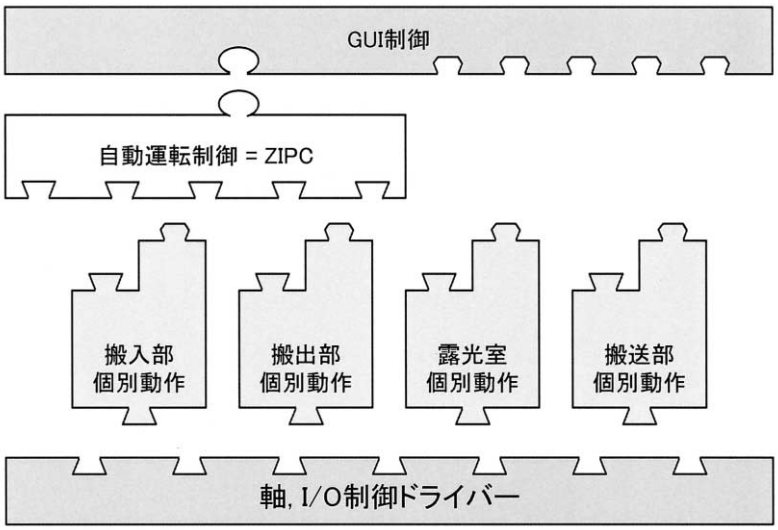
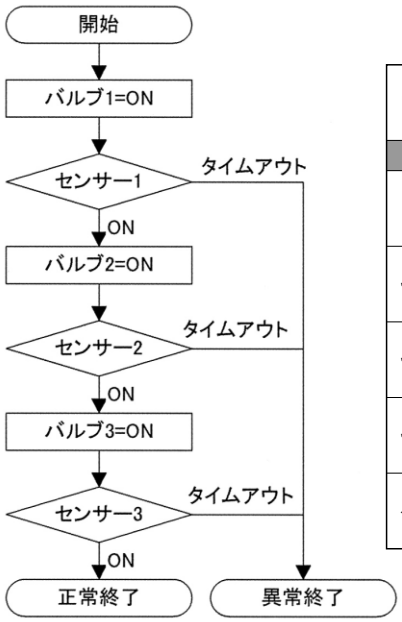


図3



□0 Watcher 9	S	初期	センサー1	センサー2	センサー3
E		0	1	2	3
開始	0	1 バルブ1をON	／	／	／
センサー1 ON	1	／	2 バルブ2をON	／	／
センサー2 ON	2	／	／	3 バルブ3をON	／
センサー3 ON	3	／	／	／	正常終了
タイムアウト	4	／	異常終了	異常終了	異常終了

図4

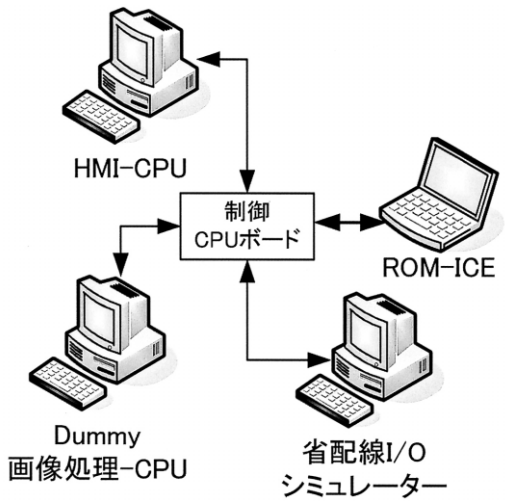


図5

使って、フローチャートどおりに動いている事を確認することとした。

もちろん、I/Oドライバーに関してはデバッグ終了後、納入することとした（図5）。

効果

状態遷移手法の優れた点や、導入効果については、既に多数の先輩諸氏が書かれているのでここでは省略することとし、実際にどういった効果が現れたかについて述べることとする。

・担当者が全体を把握しやすくなった

担当を縦割りから横割りにすることで、装置全体の動きの把握を、担当者全員が出来るようになった。

・リードタイムが短縮された

各担当ごとに設計を同時進行でき、さらに外部委託をしやすくなったことで、装置開発リードタイムが短縮された。

・早い時点で構造欠陥がわかるようになった

ZIPCで早めにシミュレーションが出来るようになったため、基本的な欠陥に早い時点で気が付くことが出来るようになった。

・実機がなくてもデバッグが可能となった

従来は、実機が組み上がるまでデバッグがほとんどできず、動かしながらデバッグせざるを得なかったため、装置組立完了から自動運転が出来るようになるまで時間が掛かっていたが、予めある程度のデバッグが終われるようになったので大幅に短縮が出来た。

・設計資料がきちんと残るようになった

納期優先により、とりあえずコーディン

グだけでも、と設計無しで手をつける（信じられないかもしれないが実際の現場ではこういう事がありがち）ことが出来なくなったため、必ず設計を行い、資料が残るようになった。

・バグを設計に戻って修正するようになった

手書きコードの場合、バグを見つけたときに、小手先のコーディングでごまかしてしまうことがあるが、ZIPCの場合は自動コーディングのため、少々面倒でもきちんと設計を変更するようになった。

・ソフトウェアの信頼感がアップした

装置メーカーの場合、機械や電装が「図面を出図」という形で、しっかりとした設計体制を採っている一方、ソフトウェアは機械/電装の不具合を解決しつつ、設計と製造を同一部署で同一担当者が行うため、「出図」という作業が行われることはまれである。よって、小手先での作業が増え、なおかつソフトウェアが目に見えないものであるため、どうしても他部署から完成度や設計そのものに対して不信感をもたれることが多い。

具体的には、不具合が発生した場合に「まずソフトがおかしくないことが分かったら、他の部分を調査する」という形になってしまうことが多々見受けられた。

しかし、ZIPCの適用と設計資料の充実により、この不信感が少しずつ和らいできたように感じている。

まとめ

このようなツールを導入する場合、最も障害になるのは現場の抵抗である。幸いにも弊社ではこの抵抗がさほど大きくなく、実績を重ねるにつれて標準ツールとなりつつある。現在販売している主力ラインナップの８割はZIPCを適用したものとなっている。

今後は更なるリードタイム短縮と品質向上（バグ削減）のため、次のようなことに挑戦中である。

・ZIPC-STMの設計外注委託

STMの資産がだんだん増えてくるに伴い、ゼロから新規に作ることが少なくなってきたので、現状あるSTMを新型機に合せて機能追加する場合であれば、ZIPCを使い慣れた依託先に依託することも可能なはずである。

・ZIPC-STMシミュレーションの充実

現在のシミュレーションでは、STMが正常に「動くこと」を確認できる。このため、正常な状態であれば装置は比較的簡単に思い通りに動いてくれる。

しかし、予想外のイベントが発生した場合などに「動かなくならないこと」を確認する手段に欠けている。

想定外の状態が発生したとき、そのステートから抜け出せず、装置が表面上ハングアップしたように見えてしまうことが時々ある。

これに関してCATS殿にご協力いただき、開発中のパーフェクトパスの新機能として、これらのチェックが出来る機能を盛込んでいただけることを期待している。