

EPICS 制御イオンポンプ電源の開発

DEVELOPMENT OF EPICS CONTROL ION PUMP POWER SUPPLY

路川徹也^{#, A)}, 山本将博^{B)}, 内山隆司^{B)}
Tetsuya Michikawa^{#, A)}, Masahiro Yamamoto^{B)}, Takashi Uchiyama^{B)}

^{A)} East Japan Institute of Technology Co., Ltd.

^{B)} High Energy Accelerator Research Organization, KEK

Abstract

Ion pumps are utilized extensively in large accelerators, such as storage rings, where ultra-high vacuum conditions are essential. However, due to the continuous application of high voltage by the power supply, frequent failures arise from issues like substrate moisture absorption, insulation failure due to dust infiltration from external sources, and corrosion. Especially in the case of equipment that has been in operation for many years, the frequency of failures increases, demanding significant costs, manpower, and time for repairs and maintenance. Additionally, external equipment control often relies on serial communication standards such as RS232C or RS485, necessitating a distinct control system. To address these challenges, we have implemented a hardware configuration designed for straightforward repair or replacement, even in the event of a breakdown. Moreover, we have integrated a single-board computer within the device, enabling remote control via Ethernet. This innovation allows for control via a web interface and EPICS, enhancing the ease of integration.

1. はじめに

蓄積リングなど大型で超高真空が必要とされる加速器では多数のイオンポンプが使用されているが、その電源は常時高圧をかけ続けるため、基板への吸湿、外部からの粉塵等による絶縁不良や腐食等で度々故障が発生する。

特に長年運用を続けてきた機器の場合、故障頻度が増え、その修理や保守には少なくないコストと時間を要するため、予備を常時十分保持しておくのが理想だが、そのためには金額的な負担が増大する問題がある。

また、市販のイオンポンプ電源の多くは外部から制御するためのインターフェイスとして RS232C や RS485 等のシリアル制御が用いられることが多く、別途制御システムが必要なことが少なくない。

これらの問題を緩和するため、壊れても修理や交換が容易なハードウェア構成にし、機器内にシングルボードコンピュータを内蔵することで、Ethernet を使ったリモート制御を可能にしたシステムを検討し、プロトタイプ機を開発した。

Web インターフェイスや EPICS[1]での制御も可能にすることで、加速器等の施設への導入が容易になることも期待できる。

2. 設計方針

保守および修理が容易・低コストで実施でき、加速器制御に容易に導入できるイオンポンプ電源を開発するために、現状の故障の主な事例と具体的な目標を検討した。

2.1 イオンポンプ電源の故障事例

これまでに多くのイオンポンプ電源の故障を経験して

きたが、代表的な故障事例としては、

- 高電圧出力部へのホコリ付着による、絶縁不良に伴う出力電圧の低下や放電による焼損。
- コンデンサ等部品の経年劣化。
- 表示パネルと出力電圧の不一致。
- リモート通信不全による制御不能。

など多岐にわたる。特に停電／瞬電後に発生することが多い。使用する環境にも依存し、特に十分な空調が行われていない場合や、粉塵などが多い環境で故障が多い状況である。

2.2 開発目標

本開発における目標は、前述の故障事例、経験より次のようなものとした。

- 電源は、7.5 kV 相当出力可能。
- 超高真空状態で常時動作させておく小出力電源と、立上げ時など真空度が悪い状態で使う大出力電源を分けることで、省電力/省資源化。
- 筐体は 19 インチラック仕様で多チャンネル。
- 粉塵付着、吸湿などによる絶縁不良が発生しにくい設計。
- 基本的な制御は全てリモートで行い、ローカルでの制御パネルは不要。
- 故障しても壊れたチャンネルだけ交換可能で、交換時には全停止してもよい(活線挿抜はしない)。
- 長期的に供給・提供が見込める部材や制御系の利用。

3. ハードウェア

3.1 高電圧部

高電圧電源モジュールには、(株)東和計測の AKT シリーズ[2]を用いることで困難な高電圧電源の設計をすることなく、高電圧出力を行えるようにした。

この電源は、密閉された筐体部から AC 入力、制御入

[#] hig-mchi@post.kek.jp

力、モニター出力の端子と、高電圧出力のシリコンケーブリングが出ており、出力電圧を可変抵抗や DAC を用いて制御することが可能となっている。電源からの出力電圧/電流値はそれぞれ電圧出力される。高圧発生部の基板は密閉された筐体内にあるため、粉塵の付着による基板の絶縁不良などが起きる頻度を抑えられると推測している。

今回使用する高圧電源は、小出力電源と大出力電源の 2 種類あり、それぞれ最大出力 10 kV/1 mA と 10 kV/15 mA となっている。出力電圧設定及び電圧電流モニター値は指定可能なので、全て 0~5 V に設定した。

高圧電源の AC 入力をリレーで制御し、電源自体の ON/OFF を行うことで、高電圧出力を ON/OFF する。

今回の試作においては、高圧コネクタは LEMO ERA.3S.410.CTL、イオンポンプは ULVAC スパッタイオンポンプ PST-200CXII を使用した。

3.2 制御部

制御用電子部品は、なるべく特殊な部品は使用せず、国内で購入可能なものを選択している。

今回使用した主要な部品は、以下の様なものである。

- 制御マイコン: ATMEGA328P or 1284P
- ADC: Analog Devices LTC2451
- DAC: Microchip MCP4725
- シングルボードコンピュータ: FRIENDLY ELEC NanoPi NEO

シングルボードコンピュータは、FRIENDLY ELEC 社の NanoPi NEO、OS は Armbian[3] 23.02.2 が動作している。

デバッグ用シリアルコンソールポートが USB で外部接続できるようにしてあり、シングルボードコンピュータの IP アドレス等の初期設定はこのポートに PC を接続して、ターミナルソフトを用いて行う設計とした。

4 ソフトウェア

使用するプログラムは、制御マイコン上のファームウェアとシングルボードコンピュータの EPICS IOC、Web インターフェイス用プログラムの 3 種類が必要なのでそれぞれ製作した。

4.1 ファームウェア

制御部の ADC/DAC 等ハードウェアを制御するファームウェアは、Arduino[4]をベースに C/C++ 言語で開発した。今回は組み込み用 OS として、ロイヤリティが発生せず Arduino 開発環境下で実装可能な FreeRTOS[5]を使用している。組み込み用 OS を用いることで、制御タスクの動作に関係なく通信が行えるようになり、上位プログラムの動作に影響が少なくなることが期待される。

4.2 EPICS IOC

EPICS IOC は、制御マイコンとの通信でデバイスを制御する中心となっている。制御マイコンとの通信には、asyn[6]と StreamDevice[7]を使用しており、EPICS レコードとして設定/取得できるようにしている。

Web インターフェイスで表示/設定されるデータも全て EPICS IOC を経由して実行されている。

4.3 Web インターフェイス

Web インターフェイスは python3 で記述されており、Framework は bottle[8]、WebClient とは REST[9] API 形式で通信を行っている。

WebClient は vue.js[10]を使うことで single page application を実現している。

WebClient からは、チャンネル毎のステータス(出力電圧/電流/温度等)と設定(出力電圧/出力 ON/OFF 等)など機器制御に必要な操作が全て行えるようにしている。

また、EPICS IOC のレコードプレフィックスが変更できるようにしてある。この機能を使用すると、レコード設定の変更と IOC の再起動が自動で行われる。

5. 高圧電源単体試験

最初に高圧電源でイオンポンプを動作できることを確認するため、簡易なテストベンチでの動作試験を行った。その際に使用したテストベンチの画像を Fig. 1 に示す。



Figure 1: 単体試験テストベンチ(試験後)。

この試験において高圧電源は、大出力仕様を1台、小出力仕様1台を使用した。出力電圧設定は 5 kV の可変抵抗で行い、電圧/電流等全モニター値は GRAPHTEC GL840 と EPICS IOC にて取得した。

20 日のテスト期間中、小出力電源、大出力電源ともにイオンポンプ用電源として問題なく動作することが確認できた。

期間中のデータを Fig. 2 に示す。

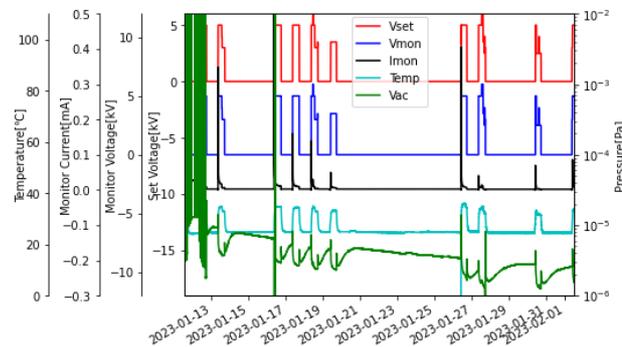


Figure 2: 高圧電源単体連続動作試験期間の履歴 (2023/1/11~2023/2/1)。

この実験の際に、小出力電源では、真空度が $1E-5$ Pa より低くならないと電圧が上がらなかった。この真空度では高圧電源の過電流保護機能が働くようだが、そのまま電圧をかけ続けても、高圧電源は故障しないことが確認できた。

6. 試作機製作と実働試験

開発は、常時使用する小出力電源を使った機器を優先して試作することにした。

基板設計は、不使用な回路が付いたまま製作してしまったが、それ以外の制御回路の動作検証に使うことにした。この回路基板には、高圧電源の AC 電源を制御するリレー回路がなかったため、別途基板を製作して対応した。

高圧電源の発熱による暴走を防ぐため、60 度で動作する温度ヒューズを高圧電源の AC 入力に挿入しており、監視のために制御マイコン内蔵 ADC とサーミスタを使った温度測定も装備した。

試作機用制御基板は、チャンネル毎に制御マイコンを搭載しており、それぞれ独立して動作するようにしたが、基板サイズの問題でケース内には 2 チャンネル分しか実装できなかった。

試作機で使用した制御回路の模式図を Fig. 3 に示す。

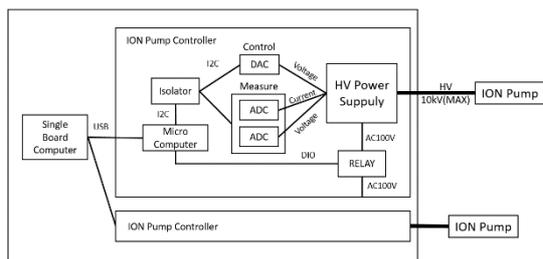


Figure 3: 制御回路模式図(試験機).

今回製作した試作機内部写真を Fig. 4 に示す。



Figure 4: 試験機ケース内部.

試作機を使用した連続動作試験は、人が対応可能な日中は 5 kV、それ以外は 3 kV を手動で繰り返し設定し、

一ヶ月以上問題なく動作することが確認できた。その試験期間中の電源電圧/電流/圧力/表面温度のグラフを Fig. 5 に示す。

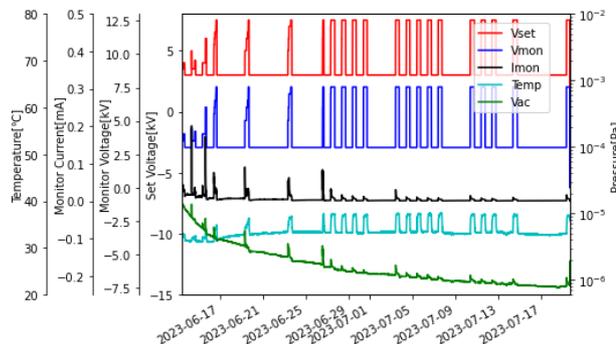


Figure 5: 試作機連続動作試験期間の履歴 (2023/6/13~2023/7/20).

6.1 消費電力測定

試作機での高圧電源単体と機器全体の消費電力を計測した。測定は、クランプメーターで AC 電流を目視で測定した結果を Table 1, 2 に示す。

Table 1: ケース全体の消費電流

設定	AC [mA]
AC リレー-OFF	33.7
CH1,2 リレー-ON(0 kV)	59.1

Table 2: 各チャンネルの消費電流

チャンネル	設定	AC [mA]
1	3 kV/1.6 μ A	25.05
	5 kV/3.2 μ A	28.6
2	3 kV/5.9 μ A	24.39
	5 kV/13.1 μ A	27.8

この結果から、各チャンネルの消費電流は平常時で AC 30 mA 以下、全体でも AC 100 mA 程度で運用できた。試作機全体でも消費電力は 10 W 程度と省電力であることが分かる。

今後制作する予定の量産機は 4 チャンネルにする予定だが、超高真空状態での運用時の消費電力は 20 W 程度に納まりそうである。

6.2 高圧電源表面温度

試験中、高圧電源単体には熱電対やサーミスタをテープで張り付けてその表面温度を測定している。

単体試験時の測定条件は、

- 測定機器: GRAPHTEC GL840
 - 側温体: K 熱電対
 - 冷却条件: 自然空冷、室内開放、ファン無し
 - 室温: 25°C程度
- 試験機での測定条件は、
- 測定機器: 制御マイコン内蔵 ADC(10bit)
 - 側温体: 石塚電子 103JT サーマスタ
 - 冷却条件: ケース内密閉、ファン無し
 - 室温: 30°C程度

それぞれ Fig. 2 と Fig. 5 にあるように、電圧出力を行うと電圧に比例して最大 5°C程度温度が上昇することを確認できた。試作機のグラフにおいてベースラインが徐々に上昇してから一定になっているのは、ケース内温度上昇と、ケースからの放熱が均衡したためだと考えられる。このことから、ケースに内蔵してもファン等の強制冷却装置は必要なく、設置個所の外気温が高めでもケースにスリット等の穴を開けるだけで十分冷却可能であると考えられる。

7. 量産試作機的设计

試作機での動作検証によって、今回製作した回路で基本的な動作には問題がないことが確認できたので、量産用システム的设计を開始した。

量産用システムは、4 チャンネルを 1 台のケースに内蔵することを基本とし、チャンネル基板と制御基板を分離することにした。ケースのサイズは高さ 3U、奥行き 450 mm の 19 インチラックサイズとした。

各チャンネル基板にはアナログ制御部、高圧電源とリレー回路だけを搭載するように変更し、チャンネル基板は I2C のみで制御するように設計し直した。

制御基板は、制御用マイコンと電源/信号アイソレータを搭載し、全チャンネルの制御と外部通信、ステータス表示を担うことにした。

制御マイコンは、プログラム容量とメモリの関係から、同じ CPU アーキテクチャの ATMEGA1284P に変更した。

SBC は、NanoPi NEO のまま変更はない。

量産試作機の制御回路模式図と完成イメージ図をそれぞれ Fig. 6, 7 に示す。

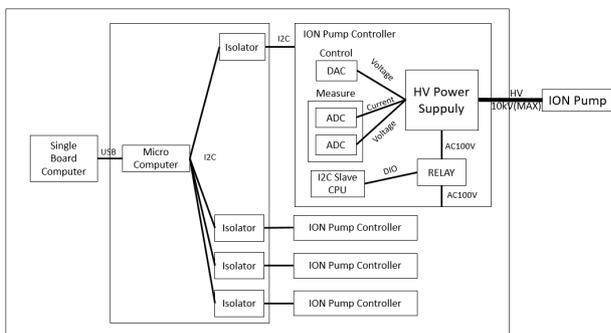


Figure 6: 制御回路模式図(量産試作機).

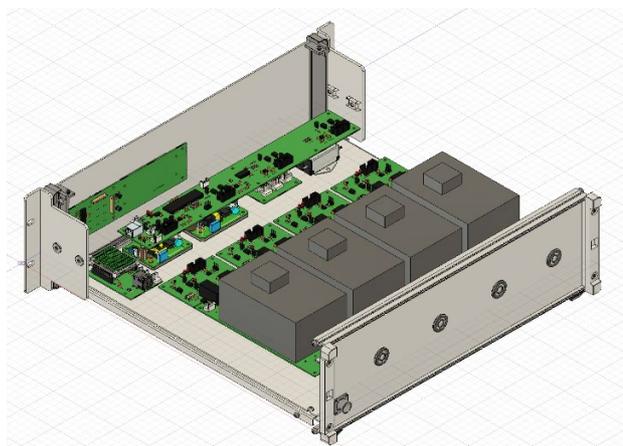


Figure 7: 量産試作機イメージ図.

8. 今後の課題

今回のイオンポンプ電源試作機的设计・開発を通して、様々な課題が出てきた。主な内容を以下に示す。

8.1 ADC 等 IC の入手性

今回 ADC に採用した LTC2451 は、昨今の半導体不足で現時点では入手が困難になっている。そのため、同一 IC でパッケージの異なる部品への変更や、多少分解能が劣るが比較的入手しやすい別の ADC での設計変更も検討する必要がある。

8.2 イオンポンプケーブル用電源側コネクタ

メーカー毎にイオンポンプ電源側のコネクタ形状が異なっており、場合によっては同じメーカーでも製造時期や型式でもコネクタ形状が異なり、電源とイオンポンプ本体の接続互換性の問題がある。コネクタおよびそれに対応する高電圧ケーブルの入手性を考慮した選択は、長期的な保守の観点からも重要である。

8.3 内蔵シングルボードコンピュータの設定

シングルボードコンピュータのデバッグ用シリアルポートから設定が変更できるようになっている。しかし、設定を行うためにはコンソールを使用した操作が必要になってくるので、IP アドレス等の設定は、ユーザーが Linux を使用できることが前提となっている。

8.4 ハードウェアインターロック・アラーム出力

真空インターロックについて、主たる真空モニターは真空計であり、現状的设计ではイオンポンプ電源からハードウェアによるインターロック出力は搭載していないが、今後必要性があれば実装を検討する。

試作機ではアラーム出力を行うようにソフトウェアは実装されているが、現状ではアラームの内容は実質温度のみである。機器内部の異常通知として必要な項目や条件の洗い出し、検討が必要になってくる。

8.5 出力装置側の接地およびケーブル接続の検出

出力先のイオンポンプが接地されていない、あるいはケーブルが接続されていない状態での高電圧出力は危

険であるため、ケーブル接続を検出する仕組みを実装する予定である。

9. 結論

市販されている高圧電源モジュールを用い、高圧電源を制御するアナログ制御部、それらを制御するマイコン制御基板、各出力チャンネルのマイコン制御基板を統括するシングルボードコンピュータで構成されたイオンポンプ電源を試作した。EPICS IOC、Web インターフェイスを実装、基本的な動作を検証した結果、長期運用でも問題なく使用できることが確認できた。

現状は試作段階であり、アラーム出力や安全インターロックの実装し、今後試験的に実働する加速器の一部分に組み込みながら必要な機能・制御を充実させ、量産に向けた開発を継続していく予定である。

参考文献

- [1] Experimental Physics and Industrial Control System, <http://www.aps.anl.gov/epics/>
- [2] 榊東和計測 AKT シリーズ, <http://www.touwakeisoku.co.jp/kt.html>
- [3] Armbian, <https://www.armbian.com/>
- [4] Arduino, <https://www.arduino.cc/>
- [5] FreeRTOS, <https://www.freertos.org/index.html>
- [6] asyn, <https://epics-modules.github.io/master/asyn/>
- [7] StreamDevice, <http://epics.web.psi.ch/software/streamdevice/>
- [8] bottle, <https://bottlepy.org/docs/dev/>
- [9] REST, https://ja.wikipedia.org/wiki/Representational_State_Transfer
- [10] veu.js, <https://ja.vuejs.org/>