LabVIEWへの統一によるQUEST中央制御システムの安定化

関谷, 泉
九州大学応用力学研究所

https://doi.org/10.15017/4482086

出版情報：九州大学応用力学研究所技術室 技術室報告. 3, pp.44-49, 2021-07. Research Institute for Applied Mechanics, Kyushu University
バージョン：
権利関係：

九州大学応用力学研究所 技術室
技術室報告 Vol. 3, 44-49

LabVIEW への統一による QUEST 中央制御システムの安定化

関谷　泉

要　旨

高温プラズマ理工学研究センターにおけるプラズマ境界力学実験装置（QUEST 装置）の中央制御システムは、主に LabVIEW と呼ばれるプログラミング言語を用いているが、一部に C 言語で記述した DLL（Dynamic Link Library）を導入している。DLL は LabVIEW のアップデートにより動作が不安定になる問題があったため、DLL を LabVIEW で書き直して言語を統一することで、システムの安定化を図った。

キーワード

中央制御システム　LabVIEW　C 言語　DLL　安定化

1．はじめに

高温プラズマ理工学研究センターでは、QUEST 装置を用いた球状トカマクプラズマの長時間電流駆動およびプラズマ-壁相互作用等の研究を行っている。QUEST 装置の内外にはさまざまな機器が設置されており、これらが連動してプラズマを発生させている。これら機器との通信（UDP）によるデータ送受信や、アナログ・デジタル信号のやりとりにより、QUEST 装置を制御するシステムが中央制御システムである。中央制御システムによって制御される機器・システムと、それらの働きを以下に示す。

- コイル用電源制御システム
  プラズマを球状に整形するコイルの電源・電流・電圧を制御する。また、過電流等の異常を監視する
- 真空排気システム
  クライオポンプの開閉と温度、真空値を監視する
- 冷却水システム
  コイル等を冷却するための冷却水を循環させ、流量・温度を監視する
- 入退室管理システム
  タッチパネルによる QUEST 本体室への入室と、扉の開閉を監視する
- ベーキングシステム
  QUEST 装置内部の高温壁の温度設定を行い、ステータス監視を行う
- サウンド端末
  設定したタイミングやエラー発生時に警告音を鳴らす

中央制御システムは、主に LabVIEW と呼ばれるプログラミング言語で記述されているが、一部に C 言語で記述された DLL が導入されている。DLL は、複数のプログラムで同時に使用できるコードとデータを含むライブラリである。DLL を利用することにより、プログラム実行速度の向上や、PC 上のディスク容量の節約が可能となる。

しかしながら、DLL の動作は、LabVIEW のアップデートによって不安定になる問題があった。そこで、この問題を解決して中央制御システムの安定化を図るため、システム内の DLL を LabVIEW で書き直して言語を統一した。なお、本検証はすべて 32 bit 版 LabVIEW 環境下で行なった。

2．中央制御システムにおける DLL の役割

中央制御システムの主言語である LabVIEW において、DLL はオレンジ色のアイコンで表示される（図 1）。アイコンの設定画面で DLL ファイルパスと呼び出す関数名を指定することにより、

LabVIEW プログラムで指定関数が実行されるようになる。

図 1　LabVIEW における DLL アイコン

中央制御システムにおける DLL の主な役割は、機器・システムとやり取りする 520 個のデジタル出入力（DIO）信号の読み取りあるいは書き込みである。また、DIO 信号に対し、表 1 に示す 7 つのパラメータを紐付けし、要素として定義する役割も持つ。これらパラメータのうち、ステータスは、DIO 信号の ON/ OFF 状態を意味する。また、固定および固定ステータスは、システム上で任意に設定でき、ステータスより優先して利用することができる（第 4 章 getValue.vi フロー図参照）。

中央制御システムで DLL が担う役割を一部例示する。

- 入力文字列と一致する信号名を持つ要素から、任意のパラメータを抜き出す
- 要素内の可変パラメータ（ステータス、固定、固定ステータス）を変更する

3．DLL から LabVIEW への書き直し

DLL の各関数を LabVIEW で書き直し、中央制御システム内の言語を統一した。また、要素に関しては、要素 No.ごとに他 6 つのパラメータをまとめて要素クラスタとし、さらに 520 個の要素クラスタを配列としてまとめ、グローバル変数として定義した（図 2）。こうすることで、中央制御システム内の各プログラム（例えば、DI 信号のリアルタイム読み込み、固定パラメータの設定など）で変更されたパラメータがただちに共有される。

図 2　要素クラスタの配列

その結果、DLL がシステムから排除され、LabVIEW アップデートで DLL の動作が不安定になる問題が解消された。

統一後、進行中の QUEST 実験に導入して試験を実施した。その結果、実験が不可能になるような問題は発生しなかったものの、中央制御用マシンの CPU 使用率が約 60%となった。統一前は 10%程度であったため、異常なほど使用率が上昇していた。

表 1　要素内のパラメータ

パラメータ	中央制御システム上の表記	型	中央制御システムにより変更	例
要素 No.	#elemNo	数値	されない	189
信号名	name	文字列	されない	bhfb_di_50
表記名	label	文字列	されない	BHFB_遠方
説明文	description	文字列	されない	BHFB_遠方
ステータス	status	ブール	される	TRUE
固定	fix	ブール	される	FALSE
固定ステータス	fix_val	ブール	される	FALSE

4．CPU 使用率の削減

QUEST 実験終了後に中央制御システムを調査した結果、CPU 使用率の異常上昇の原因は 2 つあることが判明した。それは、ループ回数が過剰であったことと、それに伴い要素の読み書きが重複していたことである。

中央制御システムにおいては、常に変化する DIO 信号をリアルタイムで読み書きする必要がある。読み書きプログラムはシステム全体で 346 個存在し、大半は 10～1000 ミリ秒の短周期ループの中に組み込まれている。それらプログラムが、520 個の要素クラスタをループごとに読み書きしていたため、CPU 使用率が上昇していた。

改善策として、不要なループ回数の削減と、要素内パラメータの取捨選択を行った。特に改善の効果が顕著であった LabVIEW プログラムである getDO.vi を例に挙げて説明する。

4-1．getDO.vi について

520 個の要素のうち、QUEST 装置内外の機器へ出力する信号（DO 信号）は 144 個である。DO 信号の状態（ON/ OFF）は、getDO.vi プログラムによって 5 行配列に成形され、5 つのモジュールを介して中央制御システムから機器へ出力され、書き込まれる（図 3）。getDO.vi のフローを図 4 に、getDO.vi 内に組み込まれている getValue.vi のフローを図 5 に示す。

ここで、getDO.vi が複雑で理解し難い構造となっている要因を以下に示す。

① 2 つのループが入れ子状態
② 内外ループの回数（図 3 における p、b）を用いた計算や条件分岐の存在
③ 条件分岐が 2 つ（1 つは getValue.vi 内）
④ 内ループの回数（b）だけビットシフトした 1 と arrayDO（5 行配列）とのビット計算

4-2．不要なループ回数の削減

前述した通り、getDO.vi は 2 つのループが入れ子状態であり、条件分岐やビット計算が含まれる複雑な構造である。しかしながら、プログラムの内容を精査すると、以下の手順で簡潔にまとめることができた。

① 520 個の要素クラスタから DO 信号に相当する部分を 144 個抜き出す
② 最小の要素 No.から順に、固定が TRUE の場合は固定ステータスを、FALSE の場合はステータスを取得する（getValue.vi 相当）
③ それらを最上位ビットから 32 個ずつ 5 行の配列に収める

上記を考慮してプログラムを改善したところ、入れ子状態が解消されてループ回数が 1 回に減り、CPU 使用率は 60%から 40%程度まで低減した。また副次的効果として、条件分岐が単純になってビット計算が排除されたため、プログラムの内容が一目で理解しやすくなった（図 6）。

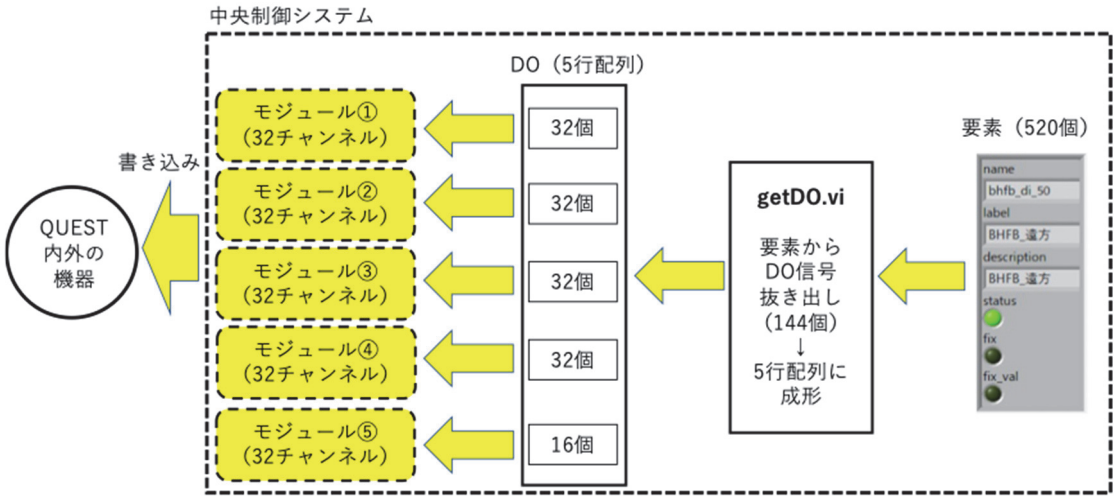

図 3　DO 信号の機器への書き込み手順

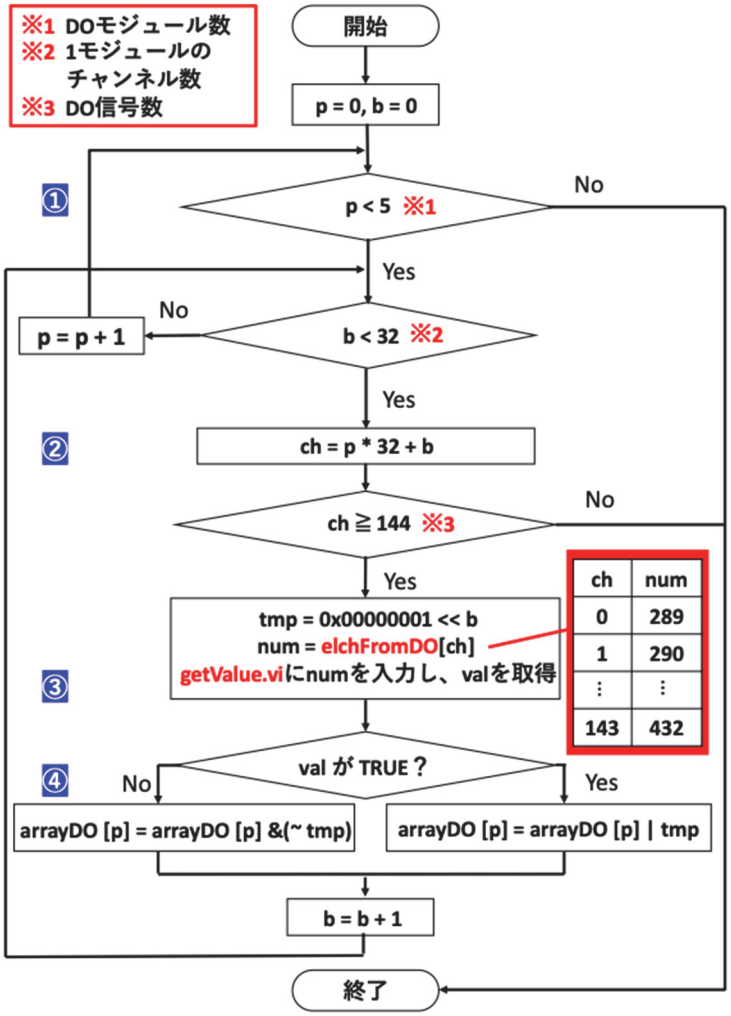

図 4　getDO.vi フロー図

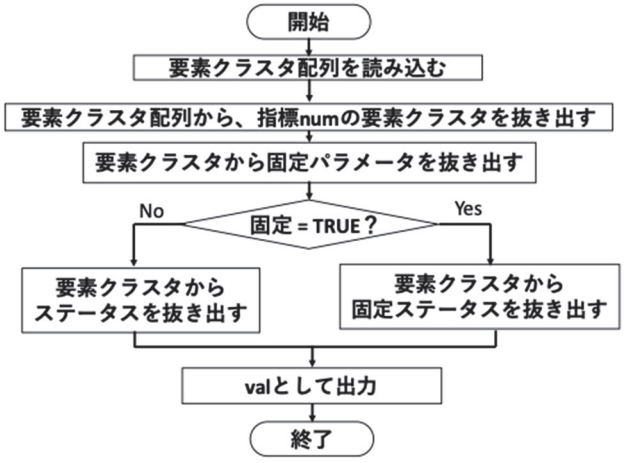

図 5　getValue.vi フロー図

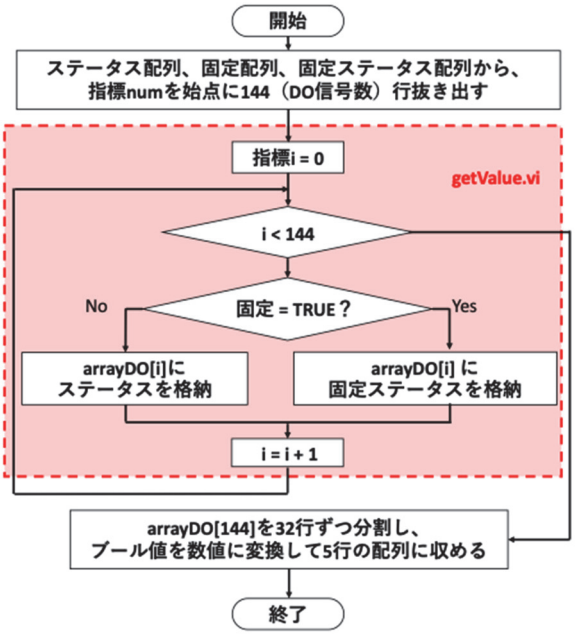

図 6　getDO.vi 改善後フロー図

4-3．要素内パラメータの取捨選択

図 7 に示すように、改善前の getValue.vi を実行すると、必ず要素クラスタ配列を読み込む。すなわち、実行するたびに 520×6 個ものパラメータをすべて読み込んでいることになり、その結果 CPU への負荷が増大していた。

改善策として、中央制御システム開始時に、6 つのパラメータをそれぞれ 520 行の配列に分割し、指標によって管理する手法を採用した（図 8)。この場合の指標は要素 No.と一致するため、指標が同一であれば同じ要素のパラメータとなる。

この手法により、中央制御システムでは変更されないパラメータ（信号名、表記名、説明文）を切り離し、プログラムから除外することができた。それに加え、各々のプログラムによって異なる必要なパラメータのみを選択することが可能になった。例えば getValue.vi で必要なパラメータは、ステータス、固定、固定ステータスの 3 つである（図 9)。

上記のような要素内パラメータの取捨選択により、CPU 使用率は 10%以下となり、DLL 使用時と同水準にすることができた（表 2)。

表 2　CPU 使用率の変化

	CPU 使用率（%）
DLL 使用	～10
LabVIEW 統一後	40～60
4-2.改善後	～40
4-3.改善後	～10

5．まとめ

中央制御システムにおいて、C 言語で記述された DLL を LabVIEW で書き直して言語を統一した。これにより、今後の LabVIEW アップデートで DLL の動作が不安定になる問題が解消され、中央制御システムの安定化を実現した。

その際、CPU 使用率が異常上昇する問題が生じたものの、ループ回数を削減し、要素内のパラメータを取捨選択して読み書き回数を最小化することで、CPU 使用率を DLL 使用時と同水準にすることができた。

6．今後

今後の QUEST 実験において、まずは 32 bit 版 LabVIEW 環境下で、改善後の中央制御システムが運用される。もし問題が発生した場合には、速やかに解決に取り組むつもりである。その後、時期は未定だが、システムを高速化するために LabVIEW を 64 bit 版に更新する予定である。

また、SN コイル電源を遠隔操作する LabVIEW プログラムの開発と、中央制御システムへの実装を依頼されている。今後も中央制御システムに関する知見を深めながら、取り組んでいく予定である。

謝辞

長谷川真助教におかれましては、QUEST 中央制御システムの改善という大役を与えてくださり、さまざまな助言をいただきました。厚く御礼申し上げます。

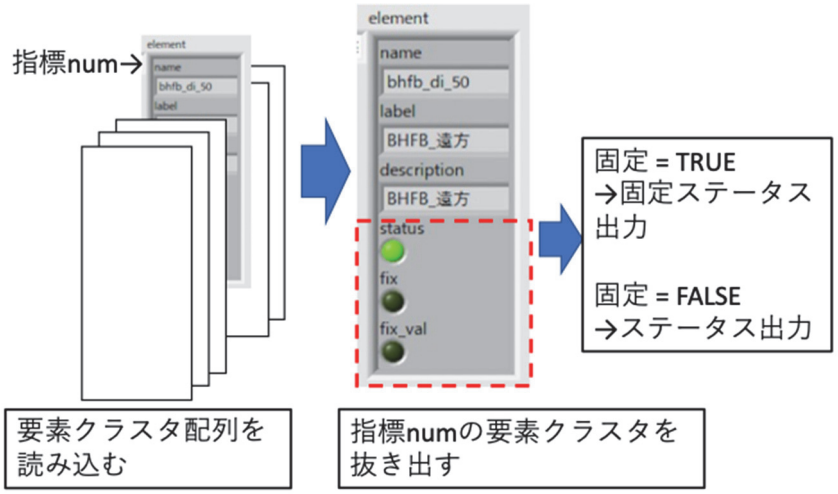

図 7　改善前の getValue.vi

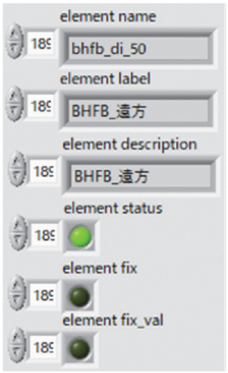

図 8　分割後の各種パラメータ配列

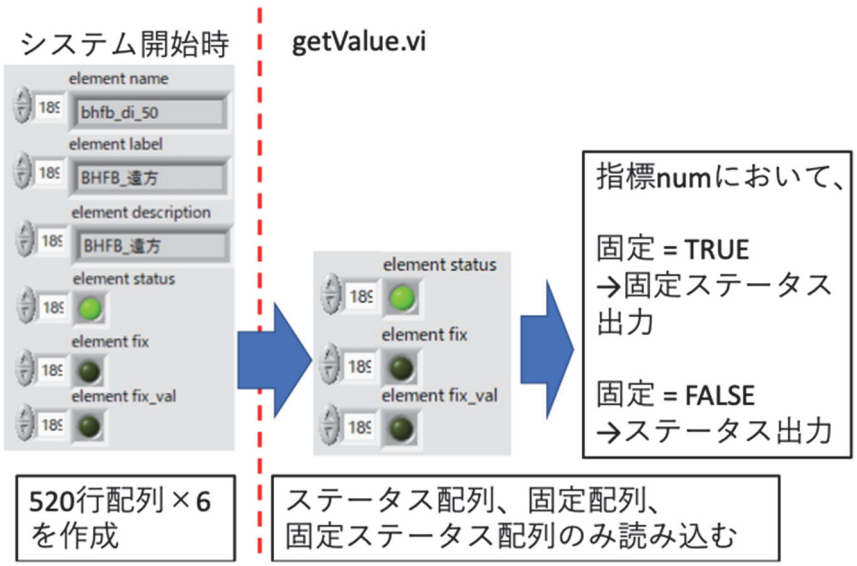

図 9　改善後の getValue.vi