Vol.2013-SLDM-160 No.33
Vol.2013-EMB-28 No.33
2013/3/14

組込みシステムへの故障対処機能追加に関する一提案

徳永 寿郎[†1]

近年の組込みシステムは安全/安定運用が必要な社会インフラに適用されつつあり，故障で停止した場合に社会に多大な影響を与える．またシステムに求められる要求も様々で複雑化しており，故障の原因も多種多様となり原因の特定や対処に時間を要してシステムダウンの時間が長くなる危険性が増大している．このような社会インフラでは，故障した場合でも原因究明が容易でシステムダウンの時間が最短で済む仕組みが必要とされ，発生した故障に対して検知/情報収集/対処するという機能(本稿では故障対処機能と呼ぶ)が重要となっている．しかし，現状では OS の機能を利用して簡易に故障対処している組込みシステムが多く，情報収集や対処が不十分な場合が多い．また，OS の故障対処機能を強化する開発は容易ではない．これらの問題を解決するため，既存の組込みシステムに対して，OS とは別モジュールの故障対処機能を容易に追加する手法について提案する．

1. はじめに

本稿では，既存の組込みシステムに対して，システムダウンを伴う故障や原因究明が必要な故障などが発生した場合に，その故障の検知/情報収集/対処を詳細に行なう機能を，容易に追加する手法について提案する．

以下，２章で背景，３章にて従来技術と課題を説明し，４章において提案する故障対処機能の方式や基本設計(構成，動作など)について説明し，５章で本提案の特徴やメリット，今後の課題事項について考察を行い，６章で今後の計画などについて述べ本稿をまとめる．

2. 背景

近年の組込みシステムは，安全が求められる車両システムや，ノンストップ運転するプラント制御システムなど，社会インフラとして重要な役割を担うシステムに適用されつつある．安全/安定運用が求められる社会インフラが停止すると社会に多大な影響を与える．例えば車両システムの制動系の場合には人命に関わる事故や交通がストップする事態の発生が考えられる．電力システムの制御系の場合には電力供給が滞る事態の発生なども考えられる．

このような社会インフラシステムでは，様々なデバイスやネットワークへの接続の要求や，開発/運用のコストの低減の要求など，システムに求められる要求も様々になっている．例えば，フラッシュメモリ，SD カード，SSD など多様な二次記憶装置への対応や，無線 LAN でのネットワーク接続や，リアルタイム処理や複数アプリケーションの並列処理などを必要とする要求もあり，組込みシステムの複雑化の度合いも増している．複雑なシステムでは故障も多種多様であり，原因特定や対処に時間を要してシステムのダウンタイムが長くなる危険性も増加している．

これらの背景により，安定/安全運用が求められ，かつ複雑性が増している社会インフラ等のシステムに適用される組込みシステムでは，原因究明が容易でシステムダウンの時間が最短で済む，故障対処の仕組み(本稿では故障対処機能と呼ぶ)が必要と考える．

3. 従来技術と課題

現状の組込みシステムでは，OS の機能を利用して簡易に故障に対処する場合が多く，故障時の情報収集や対処が不十分であることが多い．例えば，OS で故障を検知する際に，故障発生時のレジスタやスタック情報やメモリダンプなどの障害解析に役立つ詳細情報を残す仕組みや，故障の種類に応じた詳細な対処動作(LED 点灯や停止/リブート/リトライなどの最終動作など)を行なう仕組みが整っていない場合がある．このため，既存の組込みシステムに対する故障対処機能の強化が課題である．

また，既存の組込みシステムの故障対処機能を強化するには，既存の OS のモジュールの強化開発を実施する方法が考えられるが，以下の問題があり容易ではない．

(1) 故障対処はハードウェアの故障が起きている状態で対処する場合もあるため，処理が複雑であり，技術的な難易度が高い．
(2) OS の強化開発が必要で，OS の機能を利用している他の部分への影響がないことを再確認する必要がある．

複数の組込みシステムに対してこの強化開発を行なう場合には，開発工数は更に大きくなる．このため，既存の組込みシステムの故障対処機能を強化する開発を，OS の大きな変更なしに実現できることも課題となる．

†1 三菱電機株式会社情報技術総合研究所
Information Technology R&D Center, Mitsubishi Electric Corporation

Vol.2013-SLDM-160 No.33
Vol.2013-EMB-28 No.33
2013/3/14

4. 提案する故障対処機能

4.1 検討すべき課題

4.1.1 対象とする故障

本節では，組込みシステムにおける故障を分類し，それらの故障を CPU が検知する手段としての例外/割込みとの関係を整理し，本稿で対象とする故障を明確にする．この整理においては参考文献 1)を参考とした．

組込みシステムの故障には，ハードウェアに起因する故障とソフトウェアに起因する故障がある(表 1).

表 1：故障の分類と例外/割込みとの対応

故障の分類	説明	例外/割込み
(1)ハードウェアの故障		
①CPU 内のハードウェアに起因する故障		
i) 例外で検知する故障	CPU 内のハードウェアエラー(マシンチェック例外) .	○
ii)例外では検知できない故障	CPU レジスタの故障や CPU キャッシュの故障など．CPU 自身がハングアップするため例外では検知できない.	–
②CPU 外のハードウェアに起因する故障	外部割込みにより CPU に通知. CPU 外のハードウェアの故障や電源異常など.	○
(2)ソフトウェアの故障		
①例外で検知する故障	アプリケーションや OS のプログラムの実行時に CPU が検出．(0 除算，セグメンテーションフォルトなど)	○
②例外で検知できない故障	アプリケーションや OS のプログラムが独自に検知．(プログラムのバグ等で処理継続不可能となった場合など)	–

ハードウェアに起因する故障は，CPU 内のハードウェアに起因する故障と，CPU 外のハードウェアに起因する故障に分けられる(表 1 (1)①②). 更に，CPU 内のハードウェアに起因する故障は，例外で検知する故障と検知できない故障とに分けられる(表 1 (1)①i)ii)). 例えば，マシンチェック例外は例外で検知する故障である(表 1 (1)①i)). 例外では検知できない CPU 内のハードウェア故障は，例えば CPU レジスタや CPU キャッシュエラーなどであり，CPU 自体がハングアップしてしまい，例外を上げられない状態になるため，例外として検知できない (表 1 (1)①ii)).

また，CPU 外のハードウェアに起因する故障(表 1 (1)②)は，例えばネットワークカードや電源などの異常による故障であり，外部割込みにより CPU に通知される．

次に，ソフトウェアに起因する故障は，例外により検知する故障と，例外で検知できない故障に分けられる(表 1 (2)①②).

例外により検知する故障(表 1 (2)①)は，アプリケーションや OS のプログラムの実行中に CPU が検出する故障である．例えば，プログラム中で 0 による除算が実行された場合(0 除算)や，プログラムが存在しないメモリアドレスをアクセスした(セグメンテーションフォルト)場合などがある．

例外により検知できない故障(表 1 (2)②)は，アプリケーションや OS のプログラムが独自に検知するもので，例えば，プログラムのバグ等で処理継続不可能となった場合などである．

前章で従来技術および課題を提示した「OS の機能を利用して簡易に故障に対処」の部分は，OS の例外処理を利用して対処する部分に該当する．本稿では，この例外により検知する故障を対象とする．表 1 においては(1)①i)，(1)②，(2)①に該当する故障である．

また，これらの故障を検知する例外を，本稿では「異常例外」と呼ぶこととする．また，CPU の例外検知機構(後述)で検知するのは異常例外だけではなく，故障ではない例外も混在して検知する．本稿ではこの例外を「正常例外」と呼び，「異常例外」と区別することとする．

4.1.2 従来の故障対処方式と問題点

従来の故障対処方式の具体例を図 1 に示す．この例は，OS の例外処理にて，異常例外発生で検知される故障の対処を行うものであり，本節ではこの従来方式の構成と異常例外発生時の動作の概略と問題点について述べる．

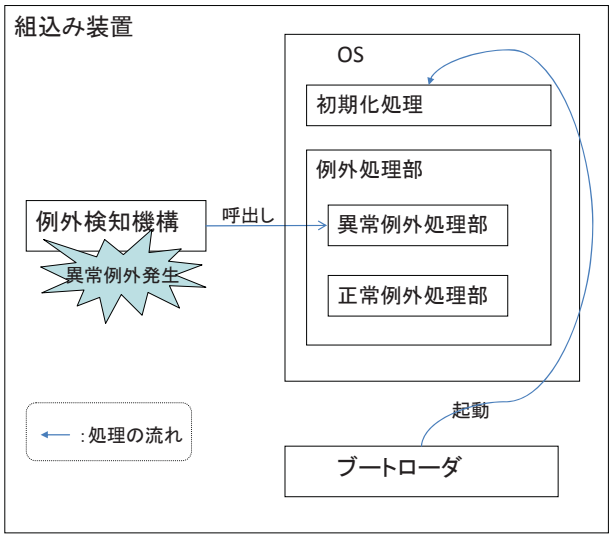

図 1： 従来の故障対処の例
(OS の例外処理で故障対処を行なう場合の概略)

Vol.2013-SLDM-160 No.33
Vol.2013-EMB-28 No.33
2013/3/14

この従来の方式の構成として，同一組込み装置内に例外の発生を検知するハードウェアである例外検知機構，独立したモジュールで構成される OS とブートローダを持つ．

ブートローダは組込み装置の立上げ時に最初に呼出されるプログラムで，OS の「初期化処理」を呼出して OS を起動する．OS の「初期化処理」は，OS を利用可能とするために，OS が用いる資源を初期化する処理である．

OS は例外に対応するための「例外処理部」を持ち，この中に，異常例外に対応するための「異常例外処理部」，正常例外に対応するための「正常例外処理部」を持つ．検知した例外が異常例外の場合は「異常例外処理部」が呼出されて故障対処を行なう．正常例外の場合には，「正常例外処理部」が呼出されて処理を行なう．

この従来の故障対処の構成に対して，新たに異常例外に対応する故障対処機能を追加して強化する場合を考えると，OS の変更，具体的には「異常例外処理部」を強化開発する必要があるが，この開発は容易ではないことが問題であることは前章にて述べた通りである．

また，他の関連技術例を見ると，異常例外に対応した故障対処機能を実現する場合，OS や仮想計算機(CPU の資源を仮想化して OS と CPU との仲介を行なうソフトウェア)に異常例外に対応する機能を追加する方法で実現している(参考文献 2))．また，参考文献 3)では，OS 側で検知した故障に対して，その対処処理を仮想計算機側で対処する技術について記述している．これらの関連技術においても，OS や仮想計算機側に故障に対処する処理を組込んで実現している．このため，既存の組込み装置にこれらの技術を含む故障対処機能を追加する場合，この変更は容易ではないという問題が存在する．

4.2 提案方式

今回提案する組込みシステム向けの故障対処機能は，故障の原因究明が容易で，システムダウンの時間が短いことを可能とする仕組みを，既存の組込みシステムに追加して強化するものである．

本方式では，異常例外による故障発生を検知し，対処方法を特定し，故障発生時の故障情報を収集し，対処を実行する仕組みを持ち，既存の組込みシステムの故障対処を強化する．本故障対処機能の処理の流れは，「故障検知」⇒「故障情報収集」⇒「故障同定」⇒「故障対処」となる(図 2)．それぞれの処理について以下に説明する．

(1) 故障検知：
異常例外として通知される故障を受付ける．

(2) 故障情報収集：
故障発生時の各種レジスタ情報やスタック情報など固有の故障情報を収集する．

(3) 故障同定：
発生した故障の特定を行い，対処方法と対応付ける．

(4) 故障対処：
故障同定で特定した対処方法に従って対処を実行する．例えば， 故障を外部に知らせるための LED 点灯，トレース情報や故障時のレジスタ等を二次記憶装置に保存するログ出力，故障時のメモリの状態を保存するメモリダンプ出力などを実行する．多重系のシステムの場合は他系への通知なども実行する．最後に，故障対処機能の最終動作(リブート，停止，呼出し元プログラムへ復帰など)を実行する．これらは，当該の故障に対応した対処方法に従って実行する．

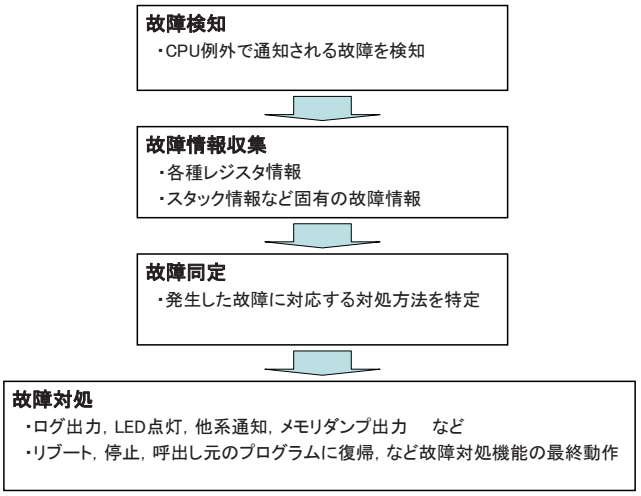

図 2： 提案する故障対処機能の概要

4.3 提案方式の設計

本節では，提案する故障対処機能の基本設計について説明する． 4.3.1 節で本方式の構成を説明し，4.3.2 節でこの故障対処機能の追加手法について提案する．また，4.3.3 節で故障対処機能追加後の故障対処の動作について，OS 動作中に異常例外発生の場合の動作と，正常例外発生時の動作に分けて説明する．

4.3.1 構成

本稿で提案する方式の構成を図 3 を用いて説明する．図 3 の構成は，図 1 で示した従来の方式(OS の例外処理で故障対処を行なう場合)と比べて，「故障対処機能」が OS とは別モジュールとして追加されているところが異なる(図 3 にて点線で囲んだ部分)．

Vol.2013-SLDM-160 No.33
Vol.2013-EMB-28 No.33
2013/3/14

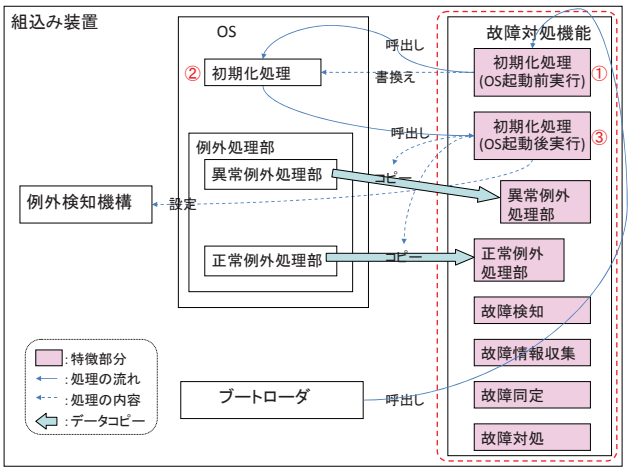

図 3： 提案する故障対処機能の構成，
および故障対処機能追加時の動作

この故障対処機能では，その初期化処理を OS の「初期化処理」よりも前に処理する「初期化処理(OS 初期化前実行)」とその後に処理する「初期化処理(OS 初期化後実行)」の２つに分けている．

また，故障対処機能側に「異常例外処理部」「正常例外処理部」「故障検知」「故障情報収集」「故障同定」「故障対処」を処理する部分を持つ．「異常例外処理部」は表 1 の(1)で説明した異常例外を処理する部分である．「正常例外処理部」は正常例外を受付ける部分である．この「異常例外処理部」と「正常例外処理部」は後述するように，故障対処機能の「初期化処理(OS 初期化後実行)」により，OS 側から必要部分をコピーしている．

4.3.2 故障対処機能追加の手法

本節では，前節にて構成を示した故障対処機能を既存の組込みシステムに対して追加する時の手法について説明する．この故障対処機能追加は，組込み装置の立上げ時に実施される．故障対処機能追加時の動作の概要は図 3 に示す．

以下，図 3 の番号①～③に沿って，故障対処機能追加の流れを説明する．

① 組込み装置の起動開始後，ブートローダより故障対処機能の「初期化処理(OS 初期化前実行)」が呼出される．「初期化処理(OS 初期化前実行)」では，故障対処機能を利用可能とするために，故障対処機能が用いる資源を初期化する．その後，OS の「初期化処理」を書換え，OS の起動完了後に故障対処機能の「初期化処理(OS 初期化後実行)」を呼出すよう変更する．次に OS の「初期化処理」を呼出す．

② OS の「初期化処理」は，OS を利用可能とするために，OS が用いる資源の初期化を実行する．その後，前述の「初期化処理(OS 初期化前実行)」にて処理が書換えられた通り，OS の起動完了後に故障対処機能の「初期化処理(OS 初期化後実行)」を呼出す．

③ 「初期化処理(OS 初期化後実行)」は OS の「異常／正常例外処理部」より必要な情報を故障対処機能の「異常／正常例外処理部」にコピーする．また，例外検知機構に対して例外発生時には故障対処機能の「異常／正常例外処理部」を呼出すように設定する．

以上の手法により， OS のモジュールの大きな変更なしに，異常例外に対応した故障対処機能を追加可能な仕組みを実現する．これにより，OS の異常例外処理以外の部分はそのまま利用できる．このため，異常例外処理以外の部分の信頼性はそのまま保持できるメリットがある．

4.3.3 故障対処機能追加後の動作

前節で追加した故障対処機能が，故障対処を行なうときの動作について説明する．

（１）OS 動作中に異常例外発生の場合の動作

OS が動作中に異常例外が発生した場合の動作を図 4 に図示する．この場合の処理フローを図 5 に示す．

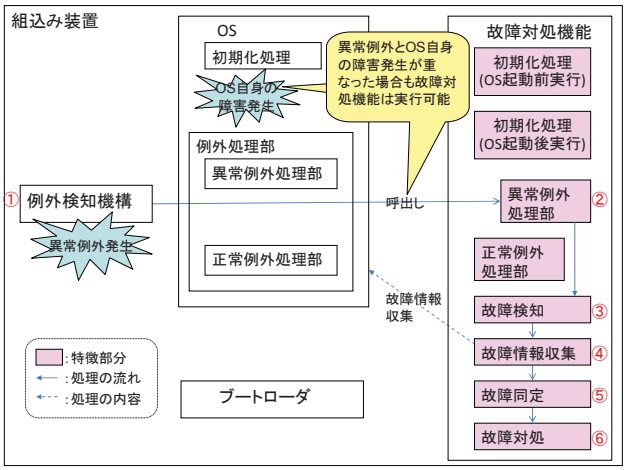

図 4： OS 動作中の異常例外発生の動作

Vol.2013-SLDM-160 No.33
Vol.2013-EMB-28 No.33
2013/3/14

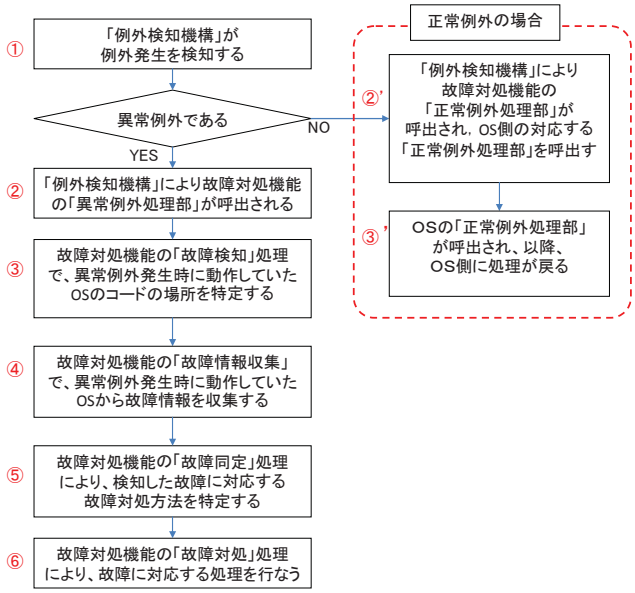

図 5：例外発生時の処理フロー

以下，図 4 と図 5 の番号①～⑥に沿って，提案する故障対処機能が，故障対処を行なうときの動作の流れを説明する．

① 例外検知機構は例外の発生を検知する．

② 異常例外の場合，例外検知機構は故障対処機能の「異常例外処理部」を呼出す．これは，前述した通り故障対処機能の「初期化処理(OS 初期化後実行)」が例外検知機構に対して例外発生時には，OS の「例外処理部」ではなく故障対処機能の「異常／正常例外処理部」を呼出すよう設定したためである．
次に，「故障検知」処理を呼出す．

③ 「故障検知」処理では，プログラムカウンタ(CPU の実行アドレスを保持しているレジスタ)を参照して，異常例外発生時に動作していた OS のコードの場所を特定する(プログラムカウンタが実行中のコードの場所より判定する)．

④ 「故障情報収集」処理では， OS から故障情報(OS のスタック情報など)を収集する．

⑤ 「故障同定」処理では，検知した故障を特定し，対応する故障対処方法を特定する．この処理は，発生した故障と，対応する対処方法の組合せを保持するテーブルを参照して実行する．

⑥ 最後に「故障対処」処理により，前段で特定された対処方法に従って故障に対応する処理を行なう．具体的な例としては，二次記憶装置(フラッシュメモリ，SD カード，SSD など)への故障情報の出力，LED 点灯や他系通知(多重系システムの場合)など，同定した対処方法に従って処理を行う．最後に，指定された最終動作(リブート，停止，元のプログラムへ復帰など)を実施する．

以上のような処理の流れとなる．

また，このように，異常例外を OS 側の「例外処理」ではなく，故障対処機能の「例外処理」で処理を行なうため，OS 自身の障害で OS が停止した場合でも，故障対処機能の「例外処理」は実行され，確実に故障対処を実施することが可能である(図 4 の吹出し部分)．

（2）正常例外発生時の動作

正常例外の発生の場合の動作を図 6 に図示する．

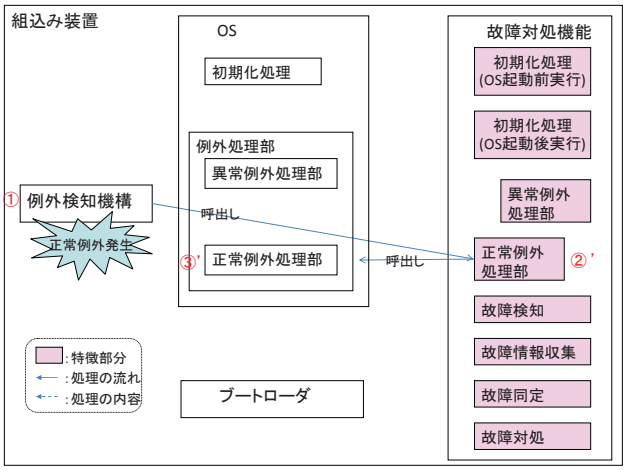

図 6： 正常例外発生時の動作

この場合の処理フローは図 5 の①⇒②’⇒③’と遷移する．以下， 図 5 及び図 6 の番号①～③’に沿って，正常例外発生時の動作の流れを説明する．

① 例外検知機構は例外の発生を検知する．

②’正常例外の場合，例外検知機構は，故障対処機能の「正常例外処理部」を呼出し，OS 側の対応する「正常例外処理部」のプログラムのアドレスを特定して OS の「正常例外処理部」を呼出す．

③’OS 側の「正常例外処理部」の処理を実行し，その後 OS 側に処理が戻る．

以上のような処理の流れとなる．

この場合，OS 側で初期化時から正常例外発生時までの間にユーザ操作などによって，正常例外の動作の登録が動的に追加・変更されていた場合でも，その追加・変更された正常例外処理を正しく実行することが可能である．

Vol.2013-SLDM-160 No.33
Vol.2013-EMB-28 No.33
2013/3/14

5. 考察

5.1 本方式の特徴／メリット

今回の方式の特徴／メリットついて考察する．本方式は以下のような特徴を持つ．

(1) 故障対処機能を OS とは独立させ，例外発生時に OS の例外処理機能ではなく，故障対処機能を先に呼びだす仕組みとしたこと．
(2) 立上げ時の初期化処理において例外処理を OS 側からコピーする仕組みとしたこと．

また，上記の特徴を持つ故障対処機能の方式により，以下のメリットがある．

(1) OS のモジュールの大きな変更なしに故障対処機能を追加することが可能となる．
(2) OS の異常例外処理以外の部分はそのまま利用できるため，異常例外処理以外の部分の信頼性はそのまま保持できる．
(3) 故障発生時の動作として，故障発生と OS 自身の故障発生が重なった場合でも故障対処機能が実行される．(OS 内に故障対処機能を内包する方式の場合, OS が停止してしまうと故障対処機能も停止する可能性がある．)
(4) 正常例外の場合の動作として，本方式では OS の例外処理部の情報を故障対処機能側にコピーして利用するが，OS 側で初期化時から正常例外発生時までの間にユーザ操作などによって，正常例外動作の登録が動的に追加・変更されていた場合でも，OS 側の例外処理部を呼出すため，その追加・変更された正常例外処理を正しく実行することができる．

5.2 今後の課題

本稿で提案した故障対処機能については，以下のようなことが今後の課題として挙げられる．

(1) 故障対処機能のモジュールを格納する静的記憶領域と，ロードして実行するメモリ領域が必要であり，適用先の既存組込みシステムに十分な空き領域がない場合には適用が困難である．従って，適用先の既存の組込みシステムの仕様を事前に調査して，本機能が追加可能かどうか判断する必要がある．
(2) 入出力エラーなどドライバがインターフェースとなる外部割込みによる故障の場合には，本方式だけでは対応できない．これについては，ドライバ側で故障検知と情報収集を行なってから，故障対処機能を関数呼出しできるようにするなど工夫する必要がある．
(3) アプリケーション側が検知した故障の場合には，本方式だけでは対応できない．これについては，アプリケーション側で故障検知と情報収集を行なってから，故障対処機能を関数呼出しできるようにするなど工夫する必要がある．

6. おわりに

本稿では，既存の組込みシステムに対して故障対処機能を強化すること，および強化する開発を OS への大きな変更なしに実現すること，という課題に対して，既存の組込みシステムを対象として OS のモジュールの大きな変更なしに容易に故障対処機能を追加可能とする手法について提案した．

本方式の特徴は，故障対処機能を OS とは独立させ，異常例外発生時に OS の例外処理機能ではなく故障対処機能を先に呼びだす仕組みであること，立上げ時の初期化処理において異常例外処理を OS からコピーすることである．

この方式によって， OS のモジュールの大きな変更なしに故障対処機能を追加可能という目的を達成できる. 更に, OS の例外処理以外の部分の信頼性はそのまま保持できること，故障発生と OS 自身の故障発生が重なった場合でも故障対処は実行可能であること，正常例外の場合は OS 側の割込み処理を呼出すので OS 側で割込み処理の動的な追加・変更があっても対応可能であること，などのメリットがある．

今後は入出力エラーなどドライバで検知する故障や，アプリケーションが検知する故障に対応するなど，本方式の課題である部分に対応した仕組みを工夫した設計を行ない，機能の開発・実装・評価を行なっていく計画である．

参考文献

1) インテル : IA-32 インテルアーキテクチャ ソフトウェア・デベロッパーズマニュアル(2004)
2) 山岸裕治，古澤かず子：特開平 01-053238(仮想計算機システムにおける本体系障害発生時の処理方式)公開特許公報，日本特許庁(1989).
3) 對馬雄次，森本俊臣，服部直也：特開 2006-155272(仮想計算機の制御方法及びプログラム)公開特許公報，日本特許庁(2006).