Vol.2016-ARC-219 No.19
Vol.2016-SLDM-175 No.19
Vol.2016-EMB-40 No.19
2016/3/24

自動車向け機能安全規格対応低負荷時間保護

月舘統宙[†1]　成沢文雄[†1]　朝倉功太[†2]　蛯名朋仁[†2]

概要：自動車制御ソフトウェアの複雑性，規模増大を受け，ソフトウェアの安全性確保のための時間保護機能が求められているが，監視対象の数や監視対象の頻度が増えると処理負荷も同時に増大することが課題である．本論文では，各故障がシステム全体に影響が発生するまでの時間間隔（FTTI）を基準とした監視間隔設計手法を提案し，それを実現する低負荷時間保護機能について述べる．本手法をエンジン制御アプリケーションに適用し，従来手法の１０％のＣＰＵ負荷で実現可能なことを確認した．

キーワード：リアルタイム OS，組込みシステム，時間保護，機能安全

Efficient Timing Protection for Automotive System compliant with ISO 26262

TSUNAMICHI TSUKIDATE[†1]　FUMIO NARISAWA[†1]
KOTA ASAKURA[†2]　TOMOHITO EBINA[†2]

Abstract: Automotive control systems have been increasing its size and complexity. To comply with the ISO 26262: functional safety standard for automotive control system, we have to implement timing protection. But it needs high CPU load depending on the number of target monitored SW and their frequency. In this paper, we propose low CPU load timing protection method that it is based on the time-span in which a fault can be present in a system before a hazardous event occurs. We evaluated the performance of the proposed method and confirmed that it needs only 10% CPU load compare than existing method.

Keywords: real-time operating system, embedded system, timing protection, functional safety

1. はじめに

自動車制御システムなどの組み込み制御システムが高度化・複雑化するのに伴い，これらを実現するソフトウェアの規模が増大し，開発効率の向上と安全性の保証が急務となっている．このような社会的背景を受け，自動車制御ソフトのソフトウェア品質レベルの統一化と安全性証明の容易化を目的に機能安全規格 ISO 26262[1]が策定された．本規格はハード・ソフトを含めたシステム全体に亘り，故障率算出方法やソフト設計手法などを規定している．

機能安全規格では車載ソフトウェアが満たすべき安全度水準(ASIL: Automotive Severity Integrity Level)を定め，規格準拠には安全度水準に応じた保護機能の適用が求めている．保護機能とは実行される処理の実行時間や実行間隔の異常を検出する時間保護機能，意図しないソフトが特定のメモリ領域にアクセスすること防ぐメモリ保護機能，送受信したデータの異常を検出するデータ保護機能である．このような保護機能はハードリアルタイム性が求められる自動車制御システムにも適用されるが，エンジン制御システムのような高頻度の割込みが発生するシステムに適用すると監視対象が動作するたびに時間保護機能が起動し負荷が増大するため適用困難である．

この問題に対して，名古屋大学 石川らは，自動車業界のデファクトスタンダードである AUTOSAR[2]が提供する AUTOSAR 標準仕様の時間保護機能を拡張し，タスクや Category2 の割り込み処理に対する時間保護機能，全割り込み禁止時間は監視しない，異常検出時はプロテクションフックではなくエラーを返すなど，機能の一部を制限することで負荷増加を低減する手法を提案している．さらに，航空機で使用される ARINC-653[3]規格をベースに，タスクの集合を 1 つの保護単位として扱い，固定周期内で各アプリケーションが実行するタイムウィンドウを決めることで監視負荷の小さい時間保護機能を提案している[4]．しかしながら，割り込みの頻度の高いエンジン制御ではタイムウィンドウの設計が難しいため適用が困難。

本論文では，システムの安全要求に応じた監視間隔を基にした低負荷な時間保護機能を提案する．システムに異常が発生してから安全状態に移行するまでの時間(以下，Fault Tolerant Time Interval: FTTI)を基準とした監視対象の監視間隔を算出することで，規格に準拠した低負荷な時間保護機能を実現した．

本論文の構成は以下の通りである．まず 2 章では背景となる FTTI ならびに時間保護機能について述べた後，従来手

†1 (株)日立製作所
Hitachi Ltd.
†2 日立オートモティブシステムズ株式会社
Hitachi Automotive Systems, Ltd.,

Vol.2016-ARC-219 No.19
Vol.2016-SLDM-175 No.19
Vol.2016-EMB-40 No.19
2016/3/24

法の課題ついて述べる．3 章では提案手法である FTTI を基準にした機能安全規格に準拠した低負荷な時間保護機能について述べ，4 章で提案手法の実装例について述べる．5 章では提案手法を自動車のエンジン制御システムに適用し評価し，6 章で結論と今後の課題について述べる．

2. FTTI と時間保護機能

2.1 FTTI：Fault tolerant time interval

機能安全規格準拠には，主機能に異常が発生した場合，異常を検出しその影響の伝播を防ぐための安全機構が必要となる(図 1 参照)．時間保護機能はこの安全機構の一つであり，監視対象である主機能に異常が発生した場合，システムに影響を与える前に異常を検出し安全状態に移行することが求められる．異常を検出し安全状態に移行するまでの時間隔内はシステムに応じて異なり，機能安全規格に準拠するには適用するシステムの安全要求から導出される時間間隔，FTTI を満たす必要がある．

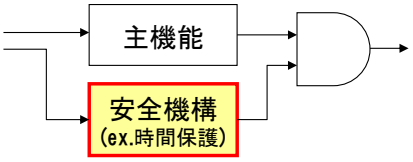


図 1 安全機構

FTTI とは，機能安全規格 ISO 26262 part 1 にて定義されている用語であり，対象システムに異常(Fault)が発生してから危険事象(Hazard)が現れるまでの時間間隔のことを指す．この時間間隔は異常が発生してから検出するまでの時間と，異常を検出してから安全状態に移行するまでの時間で構成される(図 2 参照)。FTTI の長さはシステムに応じて異なり，機能安全に準拠するには安全機構がこの時間間隔内に異常を検出しシステムを安全状態に移行させることが求められる．

例えば，FTTI が 300ms で安全状態への移行時間が 200ms かかるとすると，システムの異常を検出する機能に与えられる時間は 100msec 程度となる．

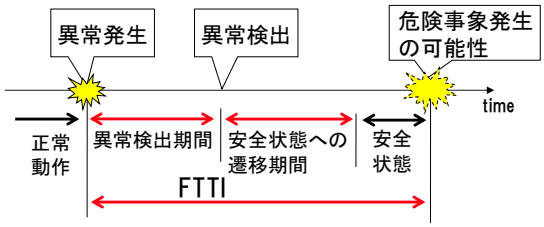


図 2 FTTI の定義

2.2 時間保護

時間保護機能の主な目的は，監視対象の処理があらかじめ設定された実行時間に対して，時間内に動作を完了しているかを監視することである．例えば，特定ソフトウェアが CPU を独占し，処理遅延やシステム機能不全が発生した場合でも，オペレーションシステムのタスクや割り込み処理の実行時間への影響(開始の遅れや実行の時間の超過)から保護することが目的である．前述した障害が発生したことを検出する手段として，実行時間の監視，デッドラインモニタリング，リソース占有時間の監視が挙げられる．図 3 に時間保護機能の例を掲載する．この例では，監視対象あるタスクの起動と終了のタイミングでタイマにアクセスすることで時間を計測しその差分から処理の実行時間を算出する．タスクが設定時間内に終了しなかった場合はデッドラインミスや実行時間違反として検出され，異常の伝播を防ぐために安全状態に移行する．このような時間保護機能は OS の機能を利用することで実現される．

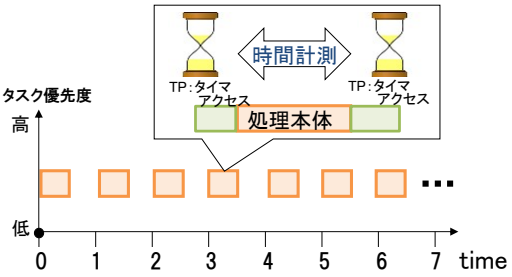


図 3 時間保護の概要

2.3 従来手法の課題

一般的に，時間保護機能を適用すると負荷が増大することが知られている．AUTOSAR[2]が提供する AUTOSAR 標準仕様の時間保護機能は，時間測定のためのタイマへのアクセス時間と，そのアクセス回数が CPU 負荷増加に大きな影響を与えるため，短い周期の処理に適用した場合大きな CPU 負荷が発生する．例えば，監視対象が 1ms，10ms ごとに起動する処理であった場合，100ms の間にそれぞれ 100 回，10 回の診断処理が実行されるため，約 10 倍のオーバーヘッドの差が生じる．そこで機能安全規格に準拠した時間保護機能を実装するための課題を以下とした．

課題： 時間保護機能は，監視対象の実行周期が短いと大きなＣＰＵ負荷が必要となる．

3. マクロ時間保護機能

3.1 提案手法概要

本論文では，提案手法をマクロ時間保護機能(以下，マクロ TP)として説明する．提案手法では時間保護機能の監視処理と診断処理の実行タイミングに着目し，従来監視対象の処理が実行されるたびに実行されていた監視処理と診断処理の実行タイミングを見直す．具体的には，監視処理と診断処理のタイミングを分離し，監視情報を基に FTTI から算出された一定の間隔で複数周期分の処理の異常診断を実行することである(表 1 参照)．これらを実現する時間保護機能を開発することでオーバーヘッドを削減し，低負荷な時間保護機能を提案する．以降，FTTI を基にした診断間隔の算出方法と FTTI を基にした監視グループの算出方法について述べる．

Vol.2016-ARC-219 No.19
Vol.2016-SLDM-175 No.19
Vol.2016-EMB-40 No.19
2016/3/24

表 1 提案手法概要

	提案手法(マクロTP)	従来手法
検出精度	監視対象全体での上限時間を指定する。通常，FTTI 以下の時間内で指定	タスク/ISRごとの上限時間を指定する
検出単位	監視対象のタスク群	個別のタスク/ISR単位
検出タイミング	監視周期ごと	異常発生直後
検出対象	主に周期実行処理	周期実行処理、イベント処理

3.2 診断間隔の算出

FTTI を基にした処理の診断間隔の算出方法について述べる．機能安全規格によると，FTTI は異常が発生してからその異常を検出するまでの時間(以下，異常検出期間)と，異常を検出してから安全状態に移行するまでの時間(以下，安全状態遷移期間)で構成される．これらの時間の合計がシステムから導出された FTTI の時間よりも小さければよい．

安全機構である時間保護機能が異常を検出するには，異常検出期間内に監視処理が 1 回以上起動し，かつ 1 回以上診断する必要がある．通常のシステムでは，発生したノイズにより意図しない動作をすることがあるが，多くの場合これらの動作は一過性の動作であり次回以降の動作に影響を及ぼさないことがほとんどである。そのため 1 度の診断で異常と判定することが難しいため，適用される安全機構に応じて複数回の監視が必用となる．与えられた時間間隔内で複数回の監視処理を実現するには通常，与えられた時間間隔よりも 1 ケタ高い精度の時間間隔で監視を実行することが望まれる．つまり，この時間間隔がシステムの安全要件から導きだされた時間保護機能の監視間隔となる。例えば，エンジン制御システムの FTTI を 1ms とし，安全状態への移行時間に 300ms かかるとすると，システムの異常を検出する機能に与えられる時間は 700ms 程度．前述した安全機能はこの時間間隔内に異常を検出する必要があるため，実際の監視間隔は与えられた時間の 1 ケタ高い精度のオーダー，数十 ms 程度の間隔となる(表 4 参照)．従来手法の監視間隔がこの時間間隔よりも短い場合，監視処理の実行頻度低減につながるため CPU 負荷の削減に貢献できる．

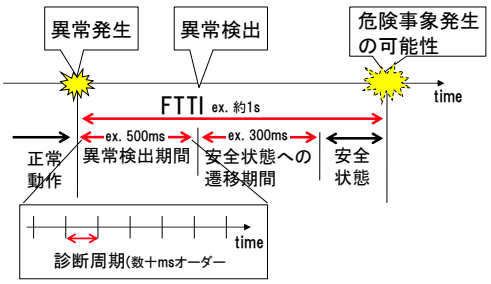


図 4 FTTI に基づいた診断間隔の算出

3.3 監視グループの形成

次に、監視グループの形成手法について述べる。全処理を 1 つの監視ソフトで監視する場合，監視グループを診断する時間間隔は，監視対象である全処理の実行周期の最小公倍数の時間間隔で形成する必要がある．算出した診断周期が前述した FTTI に基づいた監視周期(T_Detection)を超える場合，適切な時間間隔で診断できないため，監視するグループの単位を見直す必要がある．

例えば，実行周期が近い処理間のグループに分割し，各グループ内の処理の実行時間の最小公倍数をその診断グループの診断周期とする．分割した監視グループの監視周期が，FTTI を超える場合，この処理を検出時間以下になるまで繰り返すことで，FTTI を満たす監視周期と監視グループを算出する．これによりシステム安全要求を満たした時間間隔で実行される監視ソフトを実現できる．また，割り込み処理など，単位時間当たりの実行頻度が不定の場合は，従来と同様に個別の処理の実行時間の超過のみ検出することが可能．

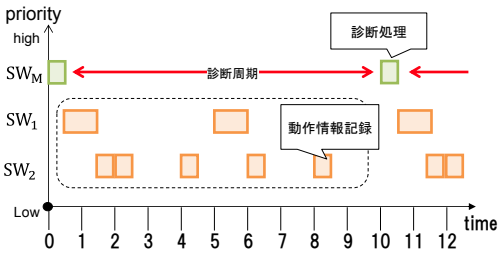


図 5 監視グループ

4. 提案手法の実装例

本章では，前章で述べた手法の実装例について記述する．前述した仕組みを実現するための仕組みとして， 4.1 時間情報監視手法，4.2 異常検出手法について記述する．

4.1 時間情報監視

監視対象となる処理の時間情報監視方法について述べる．OS が管理するソフトには一定周期で OS により起動される処理，周期処理が存在する．これら周期処理は CPU のクロックに合わせて一定の時間間隔で起動されるため，実行回数を時間情報と読み替えることが可能である．そこで，ローカル RAM に OS の周期処理に対応したカウンタを持たせ，実行されるたびにカウンタを更新する．カウンタが更新するたびに単位時間経過したことが分かるので，他の処理はタイマを直接参照しなくてもこのカウンタ値を参照することで時間情報を推定することができる．例えば，1ms で周期的に実行されるタスクにこのカウンタを設け，起動されるたびにカウンタを 1 ずつカウントアップする．カウンタ値に起動周期をかけることでこのタスクが起動されてからの経過時間がわかる．他のタスク/ISR はローカル RAM に格納されているこのカウンタの値を同様に参照することで，現在の時間を推測できる．また，基準となるカウンタの値を記憶し，一定時間後のカウンタ値との差分を取ることで，特定の基準値からの経過時間も推定できる(**エラー! 参照元が見つかりません。**参照)．この手法により，従来の

Vol.2016-ARC-219 No.19
Vol.2016-SLDM-175 No.19
Vol.2016-EMB-40 No.19
2016/3/24

タイマへのアクセスによる時間情報の取得にくらべ小さいレイテンシで時間情報を取得することができる．カウンタを更新するタスクはより高優先度のタスクによって割り込みされるため，取得できる時間の精度はカウンタを更新する周期処理の優先度と実行間隔に依存する．そのため，優先度を高く，実行間隔を短くすることにより精度を高めることができる．

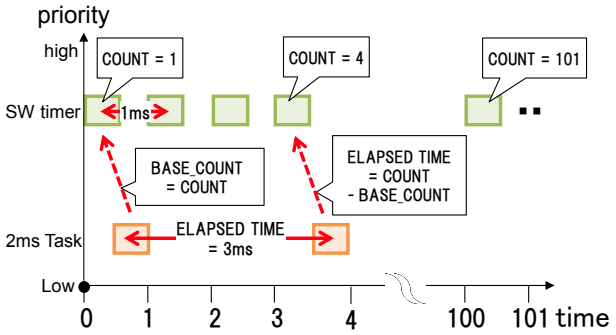


図 6 時間情報による予測

4.2 経過を用いた異常検出手法

監視タイミングを定周期化し，一度のタイミングで複数回分の処理を監視することで異常を検出する手法について述べる．従来処理の起動，終了したタイミングで時間を計測し，設定した時間や実行回数と比較することで異常を検出していた。本手法では，監視間隔を固定周期化すると，監視する処理が　起動するタイミング，終了するタイミングがわからないため従来手法をそのまま適用できない。そこで，単位時間あたりの実行回数の期待値を比較対象とする．前項で述べたように監視ソフトが周期的に実行される処理であれば，被監視ソフトは処理の実行を計測するカウンタとその処理の状態開始/終了したかを判定するフラグを持ち，監視対象の処理が実行するたびに逐次監視処理を実行すると，監視のための OS 処理のオーバーヘッドも監視処理の数だけ増加する．そこで複数の処理の監視処理を一度のタイミングで実行ことを検討する．従来，監視対象が実行されるたびに実行されていた処理を単位時間あたりの実行回数の判定に置き換えることで実現する．

4.3 提案手法の動作

提案手法の動作について述べる．システムは監視ソフトと複数の被監視ソフトで構成されるものとする．各被監視ソフトは実行されるたびに自身の動作状況をレポートとして記録し，監視ソフトは被監視ソフトの動作状況レポートをもとに被監視ソフトを診断する処理を周期的に実行するものとする(図 7 参照)．以降，被監視ソフトと監視ソフトの動作について述べる．

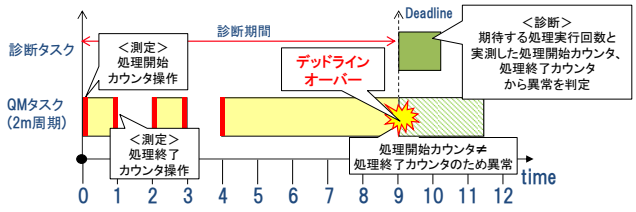
(1)被監視ソフトの動作

各被監視ソフトは，自身の動作状況を表す情報として処理の実行順番を示すシーケンス番号と処理の実行状態を判定するフラグを持つ．被監視ソフトが実行されると，まず実行判定フラグを上げ，次に開始シーケンス番号をカウントアップする．そして処理が終了する際に実行判定フラグを下げる．これにより自身の動作状況を記録する．ここで，開始シーケンス番号とは，現在の診断周期において新たに処理が開始された場合に値が更新される番号であり，線形に増加するローカル RAM に格納されるカウンタの値である．また，実行判定フラグとは，現在の診断周期において処理が実行状態に応じて更新されるフラグであり，0 か 1 の値をとるローカル RAM に格納されるフラグである．

(2)監視ソフトの動作

監視ソフトは，監視対象である被監視ソフトの動作状況を監視し，実行時間超過の検出と超過した場合におけるエラー処理を実行する．具体的には，監視対象の開始シーケンス番号が診断周期をまたいでも更新されなければ，前回実行された処理が診断周期を超過しても継続的に実行されたと判定し，エラーと判定しエラー情報を保存する．ここでエラー情報とは，アプリへの通知用に保存するログ情報と現在のタスクの正常/異常状態を表す状態情報で構成されることとする．監視割り込みソフトは全被監視ソフトを診断後，事前に設定されたエラー処理を順次実行する．ここで，監視対象シーケンス番号とは，診断周期のたびに監視ソフトが現在の開始シーケンス番号を前回開始シーケンス番号として更新した値であり，ローカル RAM に格納される．

	0	1	2	3	4	5	6	7	8	9	10	11
処理開始カウンタ	1	1	1	2	2	2	3	3	3	3	3	3
処理終了カウンタ	0	1	1	1	2	2	2	2	2	2	2	2



	期待値(許容誤差)	実測値	異常判定
処理実行カウンタ	5(±1)	3	異常
処理終了カウンタ	5(±1)	2	異常

図 7 提案手法の実装例の動作

5. 評価

本手法の有効性を確認するため，自動車のエンジン制御システムに適用し CPU 負荷率と監視に必要なオーバーヘッドを測定した．

5.1 評価環境

本手法の評価環境を表 2 に記載する．本論文ではA社製のマイクロコントローラ上に組込んだエンジン制御アプリケーションに対し提案手法を適用することで手法を評価した．OS は AUTOSAR 標準仕様である AUTOSAR OS SC3 とし，エンジン制御システムは被監視ソフトとして最大 40 個の周期処理(1ms タスクや 10ms タスクなど)とイベント処

Vol.2016-ARC-219 No.19
Vol.2016-SLDM-175 No.19
Vol.2016-EMB-40 No.19
2016/3/24

理(クランク角センサ)で構成される．提案手法はこれらの処理の実行時間や実行回数を監視することで，デッドライン違反や実行回数の異常を検出する．ここで，エンジン制御システムでは故障発生からシステムに影響するまでの時間を 800ms とし，そのための安全状態への遷移時間を 300ｍｓ，異常検出期間を 500ｍｓ，診断間隔を 30ms とした．

表 2 評価環境

評価環境	詳細
アプリケーション	エンジン制御システム
OS	AUTOSAR OS SC3
監視ソフト数	最大40
被監視ソフト数	1
FTTI	800m
安全状態への遷移時間	300ms
異常検出期間	500ms
診断周期	50ms

前章で述べた通り，被監視ソフトである周期処理及びイベント処理は自身の動作状況である実行回数と処理の状態(起動/終了)を自身が動作するたびにレポートする．監視ソフトは周期処理であるタスクや，割り込み処理である Category 2 の ISR を監視対象とする．適用システムにおける監視ソフトの優先度を，周期的に実行される QM の最高優先度の ISR よりも高い割り込み処理とすることで被監視ソフトを診断する．

5.2 評価結果

AUTOSAR 標準仕様の TP におけるタイマアクセス時間を，提案手法であるローカル RAM へのアクセスに置き換え，時間測定に必要なオーバーヘッドを概算した結果を Table 10 に記載する．評価環境である A 社製マイコンにおけるタイマアクセス時間に対し提案手法であるローカル RAM へのアクセス方法にすることで，時間保護のレイテンシが約 40 分の 1 となった．また，CPU 負荷は 10 分の 1 程度になる (表 3 参照).

表 3 結果

測定項目	提案手法	従来手法	効果 (提案手法 – 従来手法)
CPU負荷率[%]	0.37	3.63	−3.26
オーバーヘッド[us]	0.01	4.67	−4.66

マクロ TP の監視処理にかかる全実行時間の測定結果を**エラー! 参照元が見つかりません。**表 4 に示す．監視処理は，監視対象タスク/割込みの数だけ，開始シーケンスカウンタ値および実行判定フラグを読み込む処理を実行するため，監視対象タスク/割込みの数に応じて本体処理のみ実行時間が増加していることが分かる．

表 4 監視処理の負荷測定結果

監視対象の数	マクロTP前処理	本処理	マクロTP後処理	合計
Task: 0 ISR: 4	2.993us	1.439us	1.952us	6.384us
Task: 18 ISR: 4	2.992us	4.384us	1.957us	9.333us
Task: 40 ISR: 4	2.952us	7.843us	1.924us	12.719us

従来手法と提案手法の結果比較により，本手法の効果を CPU 負荷，レイテンシの観点で述べる．AUTOSAR 標準仕様の時間保護と提案手法(マクロ TP)について，時間保護ありと時間保護なしの差を，時間保護のオーバーヘッドとして Table 12 に示す．従来手法の時間保護を使用する代わりに，マクロ TP を適用することで，CPU 負荷率で−3.26 %，レイテンシで−4.66 us の性能改善効果が得られることが分かった．

6. おわりに

本論文では FTTI を基にしたシステムに応じた監視間隔と監視グループの算出方法を提案し，その監視間隔と監視グループに適用可能な時間保護機能を開発した．提案したシステムが安全状態に移行する時間を基準とした監視間隔算出手法と，複数周期分の処理をまとめて診断する時間保護をエンジンシステムに適用することで従来手法の１０％のＣＰＵ負荷で実現可能なことを確認した．今後の課題としては，提案手法が実際にスケジューリング可能か解析することである．

参考文献

[1]International Organization for Standardization: ISO 26262:2011, Road vehicles – Functional safety, 2011.
[2]AUTOSAR: http://www.autosar.org/.
[3]Avionics Application Standard Software Interface: ARINC653, 1996.
[4]石川 拓也，高田 広章, TOPPERS カンファレンス 2015: TOPPERS/ATK2 の時間保護機能拡張とりあるタイム性保障