冗長性とフェイルオーバーメカニズム

Application Note

www.isagraf.com August 2007

冗長化はプライマリシステムが故障した時にセカンダリハードウェアとソフトウェアが制御を引き継ぐことで中断無しに処理を続けることが出来る障害対策技術です。つまり、重要環境に於いて使用される制御アプリケーションの可用性を向上させる事を意味します。

現在のバージョンの **ISaGRAF** はフェイルオーバーと呼ばれるソフトとハードからなるシンプルな冗長化機構を備えています。コンフィグレーションでフェイルオーバーを有効にすると、すぐ自動的に全てのリソースが複製されます、すなわちバーチャルマシンがこのコンフィグレーションに追加されます。複製されたリソースはセカンダリターゲット上で動作する二つ目のコンフィグレーションとして追加され、他とは完全に独立します。フェイルオーバーを構成するオリジナルと二つ目のノードはそれぞれ次のように呼ばれます：

- プライマリ
- セカンダリ

Figure 1 にハードリアルタイムネットワークに接続された二つのターゲットノードにより設定されたフェイルオーバー構成を示します。

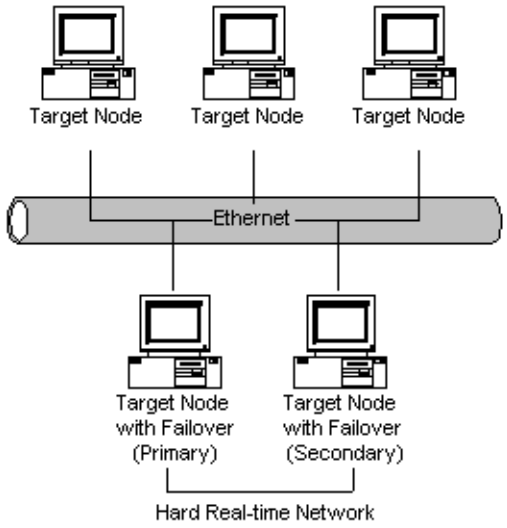

Figure 1: フェイルオーバーシステム

目次

一台のターゲットのみが常時アクティブとなり、他方はスタンバイとなります。セカンダリノードでは、プロパティはターゲット OS に依存する設定となります。QNX 6.X の場合、ノード名と IP アドレスを指定しなければなりません。Windows NT や Linux では、IP アドレスを指定しなければなりません。同期方法として次の二種類の内一方を選べます：

- パラレルポート
- I/O リンク

フェイルオーバーノード同士でハートビートの交換に使用される標準の同期方式はパラレルポートを使用します。ディフォルト値はパラレルポートベースアドレス (LPT1) で 378(16 進表示のアドレス) です。パラレルポートを使用する際には P. 6, 「パラレルポート間の接続」を参照して下さい。I/O リンクを使用する場合、ノード同士の入力と出力に用いる I/O ポイントを決めなくてはなりません。さらにアクティブとスタンバイノードの間でデータを転送するのに許される最大時間を同期タイムアウトとして設定する必要があります。どちらの通信方法も、フェイルオーバーメカニズム用のサイクル時間を指定してハートビートサイクルを定めなければなりません。

パラレルポートを使用する場合、一つのコンピュータから他方へのフェイルオーバースイッチが起きた時の最短処理時間は 5 ms です。対して、I/O リンクを使用した場合、最短時間は 500 ms です。

OS に Windows NT を使用する場合、フェイルオーバーで使用する両方のコンピュータにある system32\drivers\etc\ フォルダにある *hosts* ファイルを修正しなければなりません。127.0.0.1 mystandardip で始まる行の 127.0.0.1 を有効な IP アドレスに置き換えなければなりません。

(Windows NT プラットフォーム上で) フェイルオーバーが有効になった場合、セカンダリノードに対しワークベンチにインストールされたプログラムをダウンロードしてはいけません。

フェイルオーバーメカニズムに含まれるリソースには制限があります：

- 外部バインドは同じコンフィグレーションのリソース同士にしか設定できません
- デバイスはドライバモードで使用しなければなりません
- オンライン編集はサポートしていません
- プログラムは C ファンクションブロック内にダイナミックメモリアロケーション (malloc) や SFC ファンクションブロックを含む事は出来ません

フェイルオーバーメカニズムを設定した場合、いくつかの基本的な必要性について検討が必要です：

- ネットワークの接続速度すなわち高速イーサネット
- その通信方法での速度。パラレルポート使用は I/O リンク使用よりも性能に優れている
- プライマリ及びセカンダリノードは同じハードウエアを使用する
- プライマリとセカンダリノードは同じソフトウェア、すなわち OS, **ISaGRAF** ランタイムのバージョン，それにアプリケーションを使用

フェイルオーバー設定でトレンドを使用する場合、(プライマリとセカンダリ）どちらのコンピュータもトレンドグラフの中断を避けるためにクロックの同期をしなければなりません。TimerSync プログラムがこのサービスを提供します。プライマリとセカンダリコンピュータ各々のスタートアップスクリプトである StartRuntime.1st に以下に示す行を時間基準となるコンピュータの IP アドレスを指定した上で追加し、このプログラムを有効にします：

TimerSync -t*IPaddress* -e1000 &

これにより TimerSync がクライアントモードで起動し、周期的に指定したコンピュータの時間に合わせてローカルクロックを更新します。これはつまり指定したコンピュータでは TimerSync プログラムをサーバーモードで動作させるということです。TimerSync はなにもパラメータを指定しなかった場合サーバーモードで起動します。このプログラムに与えるパラメータのリストについては、TimerSync のドキュメントを参照して下さい。

ISaGRAF アプリケーション内でフェイルオーバーファンクションを使用すると、ターゲットノードのプロパティからフェイルオーバーに関わるたくさんの処理を行える様になります：

- フェイルオーバーメカニズムを使用するか否かを決定
- ノードをスタンバイに設定、結果他方はアクティブになる
- ノードを無期限スタンバイに設定、結果他方は無期限アクティブになる
- ノードがプライマリかセカンダリかを決定する
- ノードがアクティブかスタンバイかを決定する
- 不具合発生時に引き継ぎを行うアクティブでないノードを決定する、つまりスタンバイであるか否かを明確にしない

特定のノードが一時的にフェイルオーバーメカニズムから離れている場合、そのノードをプライマリであるかセカンダリであるかを確認しなければなりません、そのノードがアクティブがスタンバイかを確認してはいけません。ノードは常に自身がプライマリ / セカンダリの状態を保持しています。ところが、アクティブ及びスタンバイ状態は動的です。以下にどのように FAILOVER(0) コールが一つのノード上でのみ実行されるのかの例を示します。この例では二つのフラグを使用しています、一つはプライマリノードがアクティブでないことを、もう一つはセカンダリがアクティブでないことを示します。

```
who :=FAILOVER(1);
IF who = 1 AND HMI_set_standby_ind_primary = TRUE THEN
        result0 := FAILOVER(3);
ELSIF who = 0 AND HMI_set_standby_ind_secondary = TRUE
THEN
        result0 := FAILOVER(3);
END_IF;
HMI_set_standby_ind_primary := FALSE;
HMI_set_standby_ind_secondary := FALSE;
```

Getting Started（簡易版）

同じターゲット OS が動作している二つのターゲットノードが必要です：Windows NT, QNX 6.X, 或いは Linux。Windows NT や Linux ノードでは、TCP/IP ネットワークで接続されていなくてはなりません。QNX 6.X ノードでは、TCP/IP 及び Qnet ネットワークで接続されていなければなりません。これら二つのノードはパラレルポート (LPT1) 同士も専用ケーブルで接続していなければなりません。パラレルポートケーブルの仕様については page 6 に記します。

1. 使用前に LPT ケーブルが正しく接続されていることを確認して下さい。

   インストール後、ケーブルをテスト出来るようにターゲットランタイム上にテストプログラムがインストールされています。このプログラムは /usr/v2000/bin ディレクトリに置かれている TestFailOverLptCable と言うものです。Windows NT では、コマンドウィンドウで "?" を入力すればヘルプが見られます。Linux では、/usr/v2000/bin ディレクトリで TestFailOverLptCable -? と入力すればヘルプを読めます。このプログラムはディフォルトの 378h 以外のパラレルポートアドレスを使用することも可能です。

2. どちらのノードにも必要なライセンスとターゲットをインストールします。

3. ワークベンチで、コンフィグレーション一つ、リソース一つのシンプルなプロジェクトを作成します。

4. オンラインターゲット用にリソースのプロパティを変更します。

5. ワークベンチ内でコンフィグレーションに必要なネットワークを設定します：

   - Windows NT や Linux ノード用に、プライマリノードに対応した正しい IP アドレスを ETCP ネットワークに設定します。
   - QNX 6.X ノードでは、プライマリノードに対応した正しい IP アドレスとノード名を ETCP ネットワークと Qnet ネットワークに設定します。

   ワークベンチは常にプライマリノードと通信します。

6. プロジェクトを保存します。

7. フェイルオーバーエディタを開きます。

   a) ワークベンチのウィンドウメニューから *プロジェクト名* - **ハードウェアアーキテクチャ**を選びます。

   b) コンフィグレーションのタイトルバーを右クリックし、コンテキストメニューから **Advanced Options** を選びます。

   コンフィグレーション用 Advanced Options ウィンドウが表示されます。

   c) Failover タブを選択します。

8. フェイルオーバーメカニズムを設定します。

   a) **Enable Fail Over** と **Enable Parallel Port** にチェックをいれます。

   Windows NT 及び Linux ノードで、**プライマリ IP** はワークベンチで設定した値と一致させるべきです。QNX 6.X では、**プライマリ IP** と**ノード名**はワークベンチで設定したもの一致させるべきです。

   b) 冗長（セカンダリ）ノード用に必要な情報を入力します：

   - Windows NT や Linux ノード用に、**セカンダリ IP** を入力します。
   - QNX 6.X ノード用には、**セカンダリ** IP と**ノード名**を入力します。

   c) **Sync Timeout** 値としてリソースサイクルタイムの半分以下の値を入力します。ディフォルトでは、サイクルタイムは 100 ms なので、40 ms をこの **Sync Timeout**. に設定できます。

   d) まず **Apply** を、次に **Ok** をクリックします。

9. ワークベンチでプロジェクトのコンパイルとダウンロードを行います。フェイルオーバーはプライマリノードにダウンロードされ、それからプライマリノードがセカンダリを更新します、よってワークベンチはプライマリとの通信のみ行えます。

ここでプロジェクト変数を表示するための HMI を起動できます。 HMI は自動的にアクティブなマシンと接続します。 アクティブなマシンはプライマリ・セカンダリどちらにもなれ、どちらが先に起動するかに依存します。

フェイルオーバーファンクションは HMI が接続されているアクティブノードがプライマリかセカンダリかを設定できる機能を提供します。 このファンクションによりアクティブコンピュータをスタンバイに変えることも出来ます。 アクティブノードはスタンバイノードがあるか否かも確認できます。

Note: ワークベンチからのダウンロード後、転送ファイルのダウンロードによりアクティブとして使用されていたマシンはスタンバイになり、その後両方のマシンでプログラムがリスタートします。リスタートが最も速いマシンがアクティブとなるでしょう。

フェイルオーバー ST ファンクション

文法：

SINT *FAILOVER* (*operation*)

引数：

操作	SINT	フェイルオーバー処理を行う。設定可能な値： 0: ノードを無期限スタンバイに設定； ノードはリスタートしない限り決してアクティブになれない 1: ノードがプライマリかセカンダリかを表示する 2: ノードがアクティブかスタンバイかを表示する 3: ノードをスタンバイに設定； アクティブになれる 4: 他のノードの状態を表示、フェイルオーバーメカニズムの仕様； 他のノードが通常若しくは無期限いずれのスタンバイであるか 5: フェイルオーバーメカニズムが動作中か否かを表示

返り値；

結果	SINT	処理結果。結果は実行の際に与えたパラメータ "operation" に依存します。 operation = 0、ノード無期限スタンバイ設定で取りうる値： 1 = 処理成功 0 = 処理失敗 operation = 1、ノードがプライマリであるかセカンダリであるかの表示、取りうる値： 1 = ノードはプライマリ 0 = ノードはセカンダリ operation = 2、ノードがアクティブであるかスタンバイであるかの表示、取りうる値： 1 = ノードはアクティブ 0 = ノードはスタンバイ operation = 3、ノードスタンバイ設定で取りうる値： 1 = 処理成功 0 = 処理失敗 operation = 4、フェイルオーバーメカニズム仕様による他のノードの状態表示、取りうる値： 1= 他のノードは通常スタンバイ、処理の引き継ぎ可能 0 = 他のノードは無期限スタンバイ、処理の引き継ぎ不可能 operation = 5、フェイルオーバーメカニズムが動作中であるか否かの表示、取りうる値： 1 = フェイルオーバー動作中 0 = フェイルオーバー停止中

意味

フェイルオーバーメカニズムの実例として指定したターゲットノードのフェイルオーバーに関する様々な処理を実行：

- ノードをスタンバイに設定、結果として他方はアクティブになる
- ノードを無期限スタンバイに設定、結果他方は無期限アクティブになる
- ノードのモードを決定、すなわちアクティブかスタンバイか
- ノードの状態を決定、すなわちプライマリかセカンダリか
- 不具合発生時に引き継ぎを行うアクティブでないノードを決定する、つまりスタンバイであるか否かを明確にしない
- フェイルオーバーメカニズムを使用するか否かを決定

アラーム / イベントの冗長化

アラーム / イベントの冗長化にはアラームサーバー（ロガー）が動作している二台目の Windows NT マシンが必要です。ワークベンチ内で制御プロジェクトにアラーム＆イベント設定をした後で冗長化を設定します。セカンダリマシンでは、冗長化を設定する前に、間違いなく必要なライセンスをインストールして下さい。

1. アラーム＆イベントプロジェクトをプライマリからセカンダリにコピーします。このプロジェクトは **ISaGRAF** プロジェクトディレクトリのアラームフォルダに配置されています。

Note: もしディフォルト以外のディレクトリに **ISaGRAF** コンポーネントをインストールしている場合にはアラーム＆イベントプロジェクトの表示されるパスが違う場所になる可能性があります。

2. セカンダリコンピュータで、アラームサーバを起動します。

   a) スタートメニューから**プログラム**を選び、次に **ISaGRAF**、そして **Alarms** の **Alarms and Events Pro Suite Server** を選択します (ISaGRAF 以下の部分は ISaGRAF のバージョンにより名前が変わる可能性があります)。

   Alarms Server Service ControlÉEィンドウが表示されます。

   b) プロジェクトディレクトリフィールドで、アラーム＆イベントプロジェクトの場所を設定します。

   c) Alarm Service Startup オプション部分では、起動モードを選択し、**Apply** をクリックします：

   - "Manual Startup" モードを選択する場合、サーバー起動セクションの "Manual" を選択し、**Start Server** をクリック、そして **OK** をクリックします。
   - 起動モードに "Automatic" を選択した場合、**OK** をクリックしてからコンピュータを再起動します。

     アラームサーバー（ロガー）が動作します。

3. プライマリコンピュータにワークベンチ内から、冗長化を設定：

   a) ツールメニューから、**Events Logger Selection** を選択します。

   "Select Events Logger" ウィンドウが表示されます。

   b) "Secondary" セクションで、**Enable** をチェックします。

   c) アラームサーバが動作しているセカンダリコンピュータの名前かアドレスを入力し、サーバを登録します。

d) 二種類の冗長化から一方を選択します。

- **Try primary before secondary** はアラームとイベントをプライマリノードに送信し、もしこれに失敗したらセカンダリノードに送信します。 もしも後になりセカンダリノードが落ちた場合、プライマリに復帰しようとする、そのような形で動作を続けようとします。 この様な構想の下で、アラーム & イベントは一度に一台のサーバにのみ送られます。
- **Send both primary and secondary** はアラーム & イベントを常に両方のサーバに送信します。

e) **閉じる**をクリックします。

f) ワークベンチ上のプロジェクトをコンパイルしダウンロードします。

アラーム&イベント冗長化の準備がなされました。もしプライマリコンピュータ上の次にセカンダリコンピュータ上のアラーム & イベントプロジェクトを変更するならば、アラームサーバを停止し、アラーム & イベントプロジェクトを更新後アラームサーバの再起動を行う必要があります。

パラレルポート間の接続

- データポート経由でアクセスされる 5 の出力ピン – O0 から O4
- ステータスポート経由でアクセスされる 5 の入力ピン – I0 から I4
- 双方向接続

LPT1 PIN		LPT2 PIN	
O0	02	15	I0
O1	03	13	I1
O2	04	12	I2
O3	05	10	I3
O4	06	11	I4
I0	15	02	O0
I1	13	03	O1
I2	12	04	O2
I3	10	05	O3
I4	11	06	O4
18 から 25 は接地		18 から 25