## 4．デバッガー（Debugger）

新しい *HTBasic* のデバッガは、Windows 環境下で *HTBasic* を使ったプログラム開発の効率と柔軟性が最適に活かせるように設計されています。デバッガツールを使えばユーザプログラムの中の特定部分の詳細を参照することが出来ます。　デバッガを起動するには、Debug（デバッグ）┆Run Debugger（デバッガ起動）オプションを選択するか、デバッグツールバーの Run Debug（デバッグ実行）ボタン（Appendix1 参照）をクリックします。*HTBasic8.0* のデバッガのには以下の特長があります。

1）ブレイクポイント
2）ステップ機能（ステップ・イン、ステップ・オーバー、ステップ・アウト）
3）カーソルまで実行/カーソルから実行
4）用途別のデバッグウィンドウ

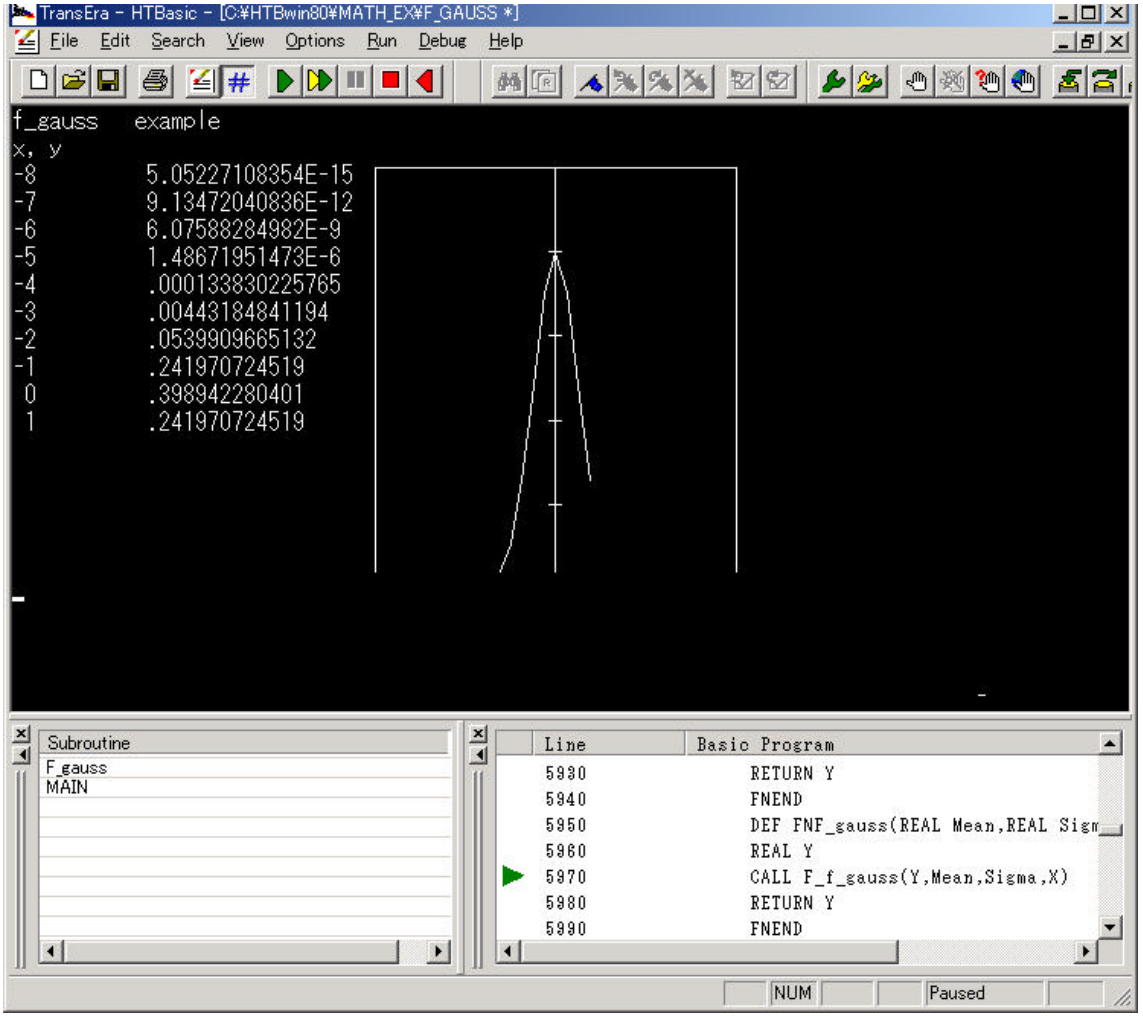


Figure2:　New Debug Window

### 1）ブレイクポイント

ブレイクポイントは、実行中のプログラムを一旦停止（ブレイク）して、その時点での変数や他のパラメーターなどの値の変化を確認するためのものです。 *HTBasic* のデバッガは、行単位、条件単位および大域でのブレイクポイントをサポートしています。行単位のブレイクポイントは、プログラムの制御が特定の行まで来たときにプログラムをブレイクします。 条件単位のブレイクポイントは、ある特定の条件が満たされた時点でブレイクします。 また大域ブレイクポイントは、制御がどこにあるかに関係なく特定の変数が指定の値になったり、特定の状況が満たされたときに、プログラムをブレイクします。

行単位のブレイクポイントを挿入するには、ブレイクさせたい行にエディタのカーソルを置き、Debug（デバッグ）メニューから Add Breakpoint（ブレイクポイントの追加）選択するか、デバッグツールバーから、Toggle Breakpoint（ブレイクポイント切り替え）ボタン（Appendix1 参照）をマウスでクリックするか、マウスを右クリックして状況メニューを表示させ、Breakpoints（ブレイクポイント）¦ Add Breakpoint（ブレイクポイントの追加）オプションを選択します。

条件単位のブレイクポイントを設定するには、Set Conditional Breakpoint（条件ブレイクポイントの設定）ダイアログを開きます。条件ブレイクを入れたい行にエディタのカーソルを置き、Debug（デバッグ）メニューから Conditional Breakpoint（条件ブレイクポイント）オプションを選択します。 または、デバッグツールバーの Conditional breakpoint ボタン（Appendix1 参照）をクリックするか、右クリックで状況メニューを表示させ、breakpoints ¦ Global Breakpoint と選択します。Set Conditional Breakpoint ダイアログが表示されたら、適切な変数、サブプログラム、条件、値等の情報を入力し OK をクリックします。

大域ブレイクポイントを設定するには、Set Global Breakpoint（大域ブレイクポイントを設定） ダイアログを開きます。 単にデバッグメニューから Global Breakpoint（大域ブレイクポイント）を選ぶか、デバッガツールバーの Global Breakpoint ボタン（Appendix1 参照）をクリックするか、右クリックで状況メニューを表示させ、breakpoints ¦ Global Breakpoint と選択します。 Global Breakpoint ダイアログが表示されたら、適切な変数、サブプログラム、条件、値等を入力し OK をクリックします。

### 2）ステップ機能（ステップ・イン、ステップ・オーバー、ステップ・アウト）

ステップ機能の 3 手法（ステップ・イン、ステップ・オーバー、ステップ・アウト）を

使用するとコードを短時間で簡単にデバックすることができます。ユーザはプログラムを通しで実行しながら、その実行内容を調べることが出来ます。

「ステップ・イン」はコードを一行ずつ進めながら、すべてのプログラムの分岐を追っていきます。
「ステップ・アウト」は、呼び出しが起こらないうちはコンテキストを最後まで続けて実行します。
「ステップ・オーバー」は、サブコンテキスト全体を実行して抜けた先のコンテキストにおいて、そのコンテキスト内の次に実行可能な行まで来たら終了します。
「ステップ・アウト」と「ステップ・オーバー」では、サブコンテキスト内の全てのブレイクポイントでブレイクします。

ステップ機能を使用するには Debug（デバッグ）メニューから Step Over（ステップ/オーバー）オプションを選びます。またデバッグツールバーから使用したいステップ機能ボタン（Appendix1 参照）をクリックするか、右クリックで状況メニューを表示させても選択できます。

### 3）カーソルまで実行/カーソルから実行

「カーソルまで実行」、「カーソルから実行」の機能を使用するとコード内を素早く柔軟に動くことが出来ます。これらの機能は、ちょうどカーソルをブレイクポイントとして設定するようなものです。「カーソルまで実行」、「カーソルから実行」は、CONTINUE コマンドと同じルールに支配されます。

「カーソルまで実行」機能を使用するには、debug(デバッグ)メニューから Run To Cursor（カーソルまで実行）オプションを選ぶか、デバッグツールバーの Run To Cursor（カーソルまで実行）ボタン（Appendix 1）をクリックします。
「カーソルから実行」機能を使用するには、カーソルからの実行をするには debug（デバッグ）メニューから Run From Cursor（カーソルから実行）オプションを選ぶか、デバッグツールバーの Run From Cursor（カーソルから実行）ボタン（Appendix1 参照）をクリックします。

### 4）用途別のデバッグウィンドウ

デバッガーツールのうち、刷新されたデバッグウィンドウ（Figure2 参照）は最強のツールの 1 つです。デバッグプログラムには、ユーザがプログラム実行中に変数やサブルーチン、ブレイクポイント、BASIC のソースコードを参照できるように、新しく 6 つのウィンドウが用意されています。

**Debug Windows（デバッグウィンドウ）ダイアログ**は View（表示）メニューから Debug Windows（デバッグウィンドウ）を選択することでアクティブになります。ダイアログが開くと、ユーザはどのウィンドウを使うかをチェックボックスを使って選ぶことが出来ます。　最初に開いた時点では、デバッグウィンドウはプログラムウィンドウの下部に配置されます。

デバッグウィンドウの位置は、プログラマの作業スタイルにあわせ配置しなおすことが出来ます。でも働きやすいような配置をしなおします。縦方向の 2 つのバー（たてが固定されている場合は、水平方向のトップバー）をダブルクリックして、ウィンドウをフローと状態にしてから、タイトルバーをクリックしてウインドウを好きな位置までドラッグします。　ウィンドウの外枠が、新しい場所に応じて変わるということに注意してください。　ウィンドウを直前にあった位置に戻すには、タイトルバーをもう一度、ダブルクリックします。2 つ以上のウィンドウがどう位置で重なっている場合は、配置されたウィンドウで制御ボタンを使うことが出来ます。このボタンを使うと、各ウィンドウのサイズを拡大・縮小することができます。

**Watch（監視）ウィンドウ**では、プログラムの各ステップ毎にユーザ定義リストで変数を登録し、プログラム実行中にこれらの値変化を参照することが出来ます。Watch （監視）ウィンドウには、Variable（変数名）、Type（型）、Value（値）の 3 つの入力欄があります。Type（型）には Array（配列）、Integer（整数）、Real（実数）、Complex（複素数）、String（文字列）、Long（ロング）、Static（スタティク）または I/O パスを設定できます。Variable（変数名）では、大文字と小文字を別に扱う（ケース・センシティブ）ことにご注意ください。

Watch（監視）ウィンドウに変数を追加するには、Debug（デバッグ）メニューから Add Watch Variable（監視変数を追加）オプションを選びます。変数を削除するには Watch （監視）ウィンドウで削除したい変数を選んでから、Debug（デバッグ）メニューから Remove Watch Variable（監視変数を削除）オプションを選択します。　または、全ての監視変数を削除するために Debug （デバッグ）メニューから Remove All Watch Variables（全ての監視変数を削除）オプションを選択します。　また、Edit（編集）ウィンドウや Watch（監視）ウィンドウ内では、マウスの右クリックを使って状況メニューを表示させ、そこから変数の追加や削除を行うことも出来ます。

NOTE

配列、スタティックおよび I/O バス変数を性格にトラック（追跡）する機能は、HTBasic

8.0 の初期リリースでは装備されませんが、今後の改定ではサポートされる予定です。

The Line Breakpoints（行ブレイクポイント）ウィンドウでは、プログラム実行に伴って、行ブレイクポイントを監視することが出来ます。このウィンドウでは、次のようなブレイクポイントのパラメータを監視します。

Enabled（オン状態）、Type（型）、Line（行）、Subroutine（サブルーチン）、Variable（変数）、Condition（条件）、Value（値）

行のブレイクポイントは、このウィンドウから個々にオンまたはオフ状態に出来ます。ブレイクポイントをハイライトし、マウスの右クリックで状況メニューをひょうじさせます。そして、Enable/Disable Breakpoint（ブレイクポイントのオン・オフ）を選択します。ブレイクポイントを削除するには、削除したいブレイクポイントをハイライトして、右クリックし、Remove Breakpoint（ブレイクポイントを削除）を選択します。ブレイクポイントを全て削除するには Remove All Breakpoints（ブレイクポイントを全て削除）を選択します。

**Global Breakpoints（大域ブレイクポイント）ウィンドウ**では、プログラム実行に伴って、大域ブレイクポイントを監視することが出来ます。このウィンドウでは Enabled（オン状態）、Subroutine（サブルーチン）、Variable（変数）、Condition（条件）および Value（値）の状態を監視します。

このウィンドウでは大域ブレイクポイントを追加することが出来ます。　ウィンドウ内で右クリックして、状況メニューを表示させ、Add Global Breakpoint（大域ブレイクポイントを追加）を選びますが、代わりにツールバーから Global Breakpoint ボタンを押したり、Debug メニューから Global Breakpoint オプションを選んでも同様のことが出来ます。

同じ状況メニューを使って、大域ブレイクポイントを個々にオンまたはオフに切り替えられます。状態を切り替えたいブレイクポイントをハイライトし、右クリックでしてし Enable-Disable Breakpoint（ブレイクポイントのオン/オフ）を選択します。

ブレイクポイント削除するには、削除したいブレイクポイントをハイライトし、右クリックして remove Breakpoint を選択します。ブレイクポイント全て削除するには、Remove All Breakpoints を選択します。

**Trace（トレース）ウィンドウ** では、プログラム中のどのコマンドが実行されているかを監視します。このウィンドウには、設定する項目は何もありません。Trace ウィンドウでは、HTBasic のコマンドラインが１つ１つ実行されるのを自動的に監視し、記録します。Trace ウィンドウにあるのは、Command（コマンドの実行記録）だけです。

Trace ウィンドウは、プログラム実行時のコマンド実行軌跡を残すという意味で、HTBasic のよく知られている TRACE 文とは本質的に異なります。 Trace 文がメッセージラインに表示する内容は、画面がスクロールされると同時に消えてしまうからです。Trace 文では、その時点で実行中のコマンドしか追跡しないことに注意してください。ブレイクポイントやサブルーチンは追跡しません。

Trace ウィンドウをクリアするには、ウィンドで右クリックし、Clear Window を選択します。

CALL（呼び出し）スタックウィンドウは、CALL スタックのビューワを開きます。（CALL スタックとは、BASIC の CALL 文で呼ばれたサブルーチンを追跡するためのものです）。このため、プログラムの実行状況を各ステップごとに監視することが出来ます。
このウィンドウで設定する項目はありません。このウィンドウでは、実行中、プログラムで定義された CALL スタックの状態を自動的に監視して、記録します。CALL スタックウィンドウにあるのは Subroutine（呼び出されたサブルーチン記録）欄だけです。

**Code ウィンドウ**では、実行中のプログラムソースコードを表示します。このウィンドウには、Line（行番号）と BASIC Code Lines（BASIC コード行）の 2 つの欄があります。これらの欄には、BASIC のソースコード内で行番号とその行番号に該当する実際のコード内容とが表示されます。このウィンドウでも、ブレイクポイントやブックマーク、プログラムポインタ（コンピュータのコード実行開始位置を性格に示すポインタ）を見ることが出来ます。

Code ウィンドウは、Trace ウィンドウが実行時のプログラム行を 1 ステップずつ進み、実行がすんだコード行だけを表示するという点で、Trace ウィンドウとは異なります。Code ウィンドウでは、ビューワでソースコード内を前後しながら参照することが出来、このソースコード内にまだ実行前のコード行が含まれていてもよいのです。また、Code ウィンドウはデバッグ時、BASIC のソースコードプログラム内のブレイクポイントの参照や修正に使うことが出来ます。