enjoy Development

KMC

SOLID

RTOSと先進のClangコンパイラを
一体化した開発プラットフォーム

開発準備から性能解析までをシンプルに変える

- オープンソース ITRON TOPPERS
- 最新のLLVM / Clang コンパイラ
- アドレスサニタイザのデバッグ支援機能
- Visual Studioベースの統合開発環境

京都マイクロコンピュータ株式会社

http://www.kmckk.co.jp/SOLID/

LLVM/Clangの可能性にゾクゾクした

SOLIDという新しいプラットフォーム。"SOLID開発の中心的メンバーに、SOLIDの意図や目的、今後の進化について聞いた"

SOLID 担当エンジニア
佐藤 大介

はじまりはLLVM*

初めてLLVM/Clangを知ったのはいつですか？

入社1,2年目の2008年頃、イリノイ大学の中でオープンソースソフトウエアであるLLVMが研究されていて、日本のオタクの間でも盛り上がっていました。当時LLVMに関してLife Long Optimization（いつでも最適化できる）という凄い目標の論文が出ていたのでそれに大変興味を持ち、カンファレンス等に参加するようになりました。LLVMの活動に注目していたところ、レポジトリの開発者メールアドレスがいつの間にか@apple.comに変わっていて、「あ〜、この人Apple社に入ったんだなぁ」と思っていたら、LLVMがiOSの開発環境として採用されたんです。Clangプロジェクトがオープンソース化されたのは2007年ですが、Clangの名前が知れ渡って急激に広まってきたのはそのころ、2010年頃からじゃないかと思います。

LLVM/Clangのどこが凄いと思ったのでしょうか？

きれいにモジュール化されて作られているオープンソースなので、各々の機能をブラッシュアップしやすいし、新しい機能追加が比較的容易にできるところです。僕らの得意分野であるテスト支援機能やデバッグ支援機能の強化等も、自分たちでできますからね。またオープンソースのコミュニティに主要な企業が参加している点でも、更なる進化が期待できると思います。ちなみにARMの64bit版コンパイラは、ARM社の人たちが最初から参加してLLVM/Clang版が開発されました。

開発者の悩み、ツールベンダーの覚悟

LLVM/Clangのポテンシャルの高さは開発者に恩恵をもたらすことができますか？

今お話しした機能拡張に対する自由度の高いLLVM/Clangを実際にソフトウエア開発者が利用する場合、現実には「ふたつの壁」が存在します。ひとつは、開発者は限られた時間、限られたリソースで早く確実に製品をリリースしなければいけないので、使い慣れた実績のあるツールを変えるリスクをとりづらいことです。ふたつめは、LLVM/Clangの持つ解析機能を使いこなすには、開発者自身がツールの仕組みを理解したうえでプログラムに解析機能を有効にするためのコードを埋め込まないといけないところです。開発者が本来目的としている機能以外のコードを書く手間をかけられるか、というとそれは難しい。

ツールベンダーとして開発環境をよくするために、“コードを埋め込む”という課題にも取り組んだのですね？

はい、やはり僕らはツールのプロですから、僕らが責任もってコードを埋め込む。そこに一歩踏み込む覚悟で臨まなければいけないと思いました。そして実装の点では、例えばテスト機能のアドレスサニタイザ**のようなエージェントを、予め動いている状態で開発者が使えるように、ツールとOSを一体化したプラットフォームを丸ごと提供することにしました。開発者はツールのデバッグ支援機能の使い方を習得するのではなく、使いたい機能を選ぶだけ。あとはツールが自動的に組み込んだデバッグ支援機能が、プログラム実行中に不具合箇所をどんどん教えてくれる。困る前にツールがミスを教えてくれる、そんなクールな仕掛けを作り込むことができました。

社内で反対意見などはありませんでしたか？

LLVM/Clangに注目し始めたころから、社内では「面白そうだから、よ〜く見張っておけ」という雰囲気はずっとありました。オープンソースであるLLVM の機能を、当社のデバッガに組み込んだ実績もあります。SOLID-IDEとしてVisual Studioを導入するときも、今までの当社のデバッガの「顔」と大分違うので、ベテランの（笑）開発者からはとっつきにくいという反応もありましたが、それも使いやすさを体験しているうちに皆推進派に変わっていきました。それにVisual Studioはゲーム業界のプログラマーの間ではデファクトになっていたという背景も後押ししたと思います。イケてるIDEだな、と。

CPU性能、メモリ容量のボーナスを品質向上に活用する

アドレスサニタイザのようなデバッグ支援機能を使うと、ユーザープログラム以外のコードが走ったり、メモリを余計に消費しますが、それに対する懸念はありませんか？

そこは、あまり問題にならなくなってきていると思います。実際にCPUの動作周波数は日々高速化していますし、使えるメモリ容量も大きい。目標性能に対する余裕があるなら、その余裕を品質向上や、日程通りに開発するという最も重要な課題に使うべきではないでしょうか。

SOLIDは進化し続けるのですね？

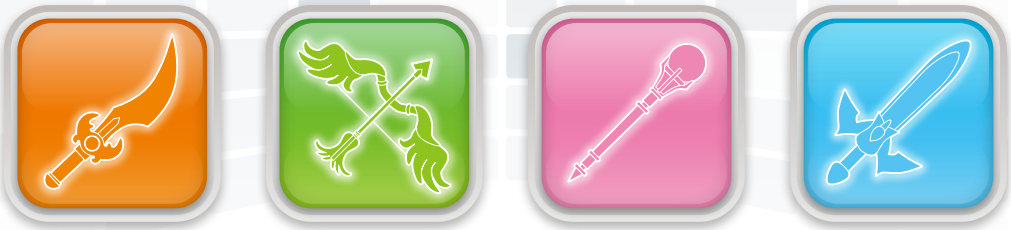
はい、まだ計画中の開発支援機能がたくさんありますので、楽しみにしていてください。もちろん、開発者の方々から「こんな機能はできないのか？」というご要望をいただければ、ツールベンダーとして知恵を絞って実現していきたいと思います。

LLVM/Clangを選んだのは、そんな将来性が楽しみだからなんです。まあ、最近はコミュニティが活発過ぎて、メーリングリストとか追っかけるのが大変になってきてるんですけどね（笑）

* LLVM: Low Level Virtual Machine
** アドレスサニタイザ: アドレスエラーを検出する、ユーザーコードに組み込まれたライブラリ（文中の社名、製品名は各社の商標です）

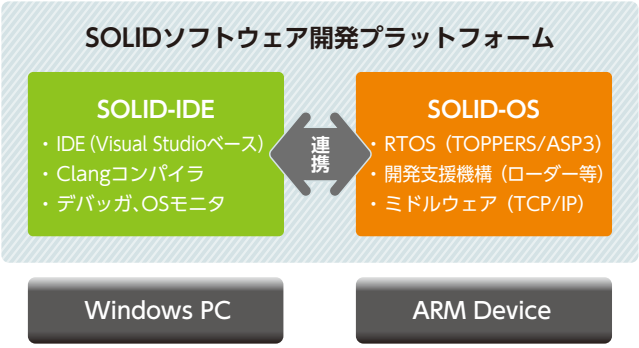
SOLID

RTOSと先進のClangコンパイラを一体化した開発プラットフォーム



RTOSと先進のClangコンパイラを一体化した開発プラットフォーム

ソフトウェアプラットフォームSOLIDは、ARM®プロセッサで動作するリアルタイムOSであるSOLID-OSと、Windows®パソコン上で動作する統合開発環境SOLID-IDEから構成されます。



SOLID-OS

オープンソースITRON仕様のTOPPERS/ASP3カーネルを基盤とし、独自にメモリ管理機能やローディング機能の対応などを強化しています。

SOLID-IDE

Visual Studioをベースに開発した統合開発環境で、エディタやビルダーそしてデバッガを含んでいます。SOLID-IDEには実績が豊富なKMC製GCCコンパイラ（exeGCC）の他、新規にLLVM/Clangコンパイラを採用しています。

LLVM/Clangコンパイラ

ARM コンパイラ6でも採用されている、先進的なオープンソースソフトウエアであるLLVM/Clangコンパイラを採用しました。

デバッガ

SOLID-IDE に統合された専用デバッガで、通常のソースデバッグやSOLID 専用のデバッグ機能が使えます。デバッグ接続は、PARTNER-Jet2が使える他、ユーザーシステムにカスタマイズしたオンラインモニタも利用可能になっています。

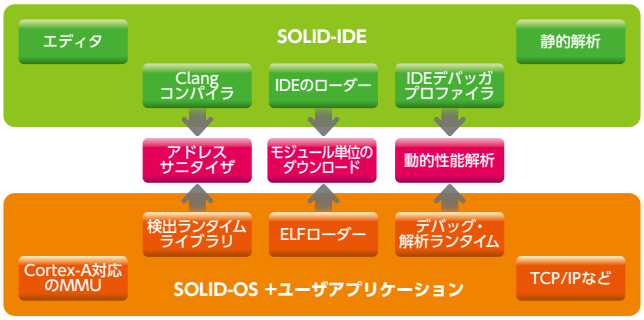
一体化することでツールは劇的に「かしこく」「スマート」になる

IDEとRTOSの一体化で便利になること

SOLIDプラットフォームでは、SOLID-OSにデバッグ支援用として組み込んだランタイム（エージェントファームエア）を、SOLID-IDEが持つ充実した解析機能と連携させています。このようにIDEとRTOSを一体化することにより、例えばiOSアプリケーションで使われているXcode開発環境のアドレスサニタイザ機能を実現しています。

MMUをRTOSで有効に使う

SOLIDプラットフォームのターゲットはARM Cortex-Aデバイスです。SOLIDプラットフォームではMMU対応のベアメタルローダーを新規開発、ローダーがELFのヘッダを参照し、仮想・物理アドレスのMMUマップを判断し物理空間に効率的にロードします。



enjoy Development エンジニアのために

SOLIDの開発ツールは、エンジニアの方々の開発のしやすさ、を目標に設計しています。そして、開発のしやすさが、エンジニアの方々の開発の楽しさにつながるよう、SOLIDは enjoy Development のコンセプトを実現できるようデザインしております。ぜひこの新しい開発スタイルを体験してください。

組み込みシステムの準備段階からプログラミング・デバッグまで、開発者の快適さを追求した、新しいプラットフォームです。

準備

OS・カーネル設定　コンパクトで実績の高いTOPPERSカーネル

- 名古屋大学を中心としたTOPPERS プロジェクトで開発されたオープンソースITRONであるTOPPERS/ASP3をSOLID-OSに採用しました。
- 組み込み機器として実績のあるμITRON 4.0仕様に準拠しています。
- ASP3のティックレス仕様により実行効率・電力効率が良くなります。
- カーネル本体、プロセッサ依存部、代表的なボードのBSPを提供します。
- ロイヤリティフリーです。

センサ・メカ制御に強いRTOSであるTOPPERSを採用し、TCP/IP標準添付でネットワークアクセスも可能

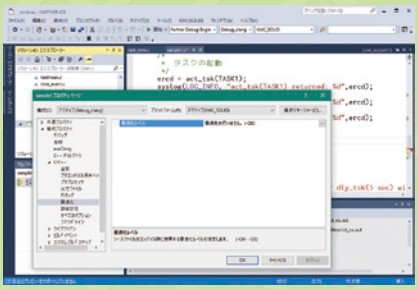
ミドルウエア設定　ミドルウエアの実装は、ツールにお任せ

- TCP/IP, Fileシステム、スクリプトエンジン等のミドルウエアを提供。
- パートナー企業製のミドルウエアの実装・連携。

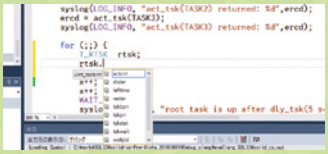
作る

コーディング　Visual Studioベースの IDEで軽やかに

- SOLID-IDEの操作画面はユーザーインタフェースに定評のあるMicrosoft社のVisual Studioをベースに京都マイクロコンピュータが独自開発したロイヤリティフリーのIDE（統合開発環境）を採用しました。
- コンパイルからデバッグまで、全てWindows上で作業できるので、ソースコード全体を見渡すことができ、「ビルド＞転送＞実行＞デバッグ」、の手順がシンプルに行えます。
- Intellisense機能を備えたコード補完型のエディタで、快適なタイピング環境を提供します。



Visual Studio ベースで作成されている、SOLID-IDE

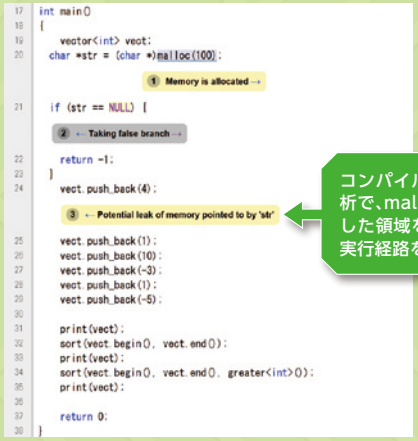


エディット時のインテリセンス機能

動画で詳しく見る

コンパイル　LLVM/Clangコンパイラの解析機能を使い倒す

- SOLIDプラットフォームでは、静的解析ツール、動的解析ツール機能を豊富に備えたLLVM/Clangコンパイラを採用しました。
- Clangコンパイラは、オプションや言語仕様拡張など、GCCコンパイラとの互換性が高い仕様です。
- ビルド時に静的解析ツールとしてClangを使用することにより、「未初期化変数の利用」「メモリリーク（解放もれ）パスの検出」などが検出可能です。
- Clangコンパイラで検出したエラーを、IDE上で分かりやすく表示します。



コンパイル時の静的解析で、malloc() で確保した領域を開放しない実行経路を検出した例

組み込みシステムの準備段階からプログラミング・デバッグまで、開発者の快適さを追求した、新しいプラットフォームです。

組み込み初登場！ アドレスサニタイザ

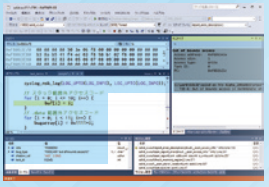
アドレスサニタイザは、LLVM/Clang コンパイラのデバッグ・テスト支援機能の一つで、メモリ破壊やリークなどを実行時に検出する動的解析機能です。使い方は簡単。アドレスサニタイザモードでビルドして実行するだけです。自分で考えてブレークポイントを設定するなど必要ありません。SOLID-IDE とSOLID-OS が連携して、間違ったメモリアクセスを自動的にあぶり出してくれます。

バグ付きのソースコード

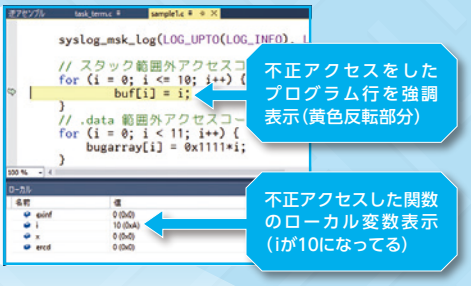
```
static unsigned short bugarray[10];

void root_task(intptr_t exinf)
{
        int             i, x = 0;
        ER              ercd;
        char            buf[10];

        // スタック範囲外アクセスコード
        for (i = 0; i <= 10; i++) {
            buf[i] = i;
        }
        // .bss 範囲外アクセスコード
        for (i = 0; i < 11; i++) {
            bugarray[i] = 0x1111*i;
        }

        /*
        * タスクの起動
        */
        ercd = act_tsk(TASK1);
```

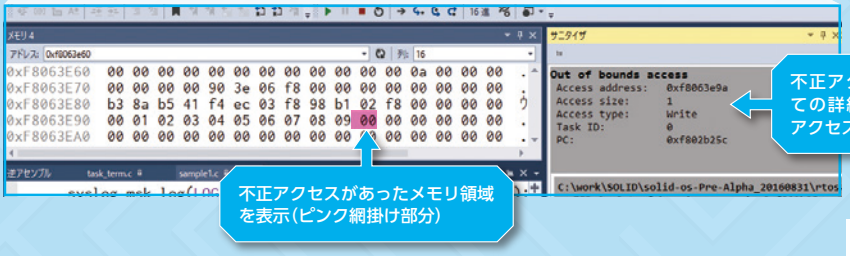
10個の配列サイズを越えた書き込みが、i=10の時に発生する

このようなメモリ破壊バグは、破壊される領域によっては、特定条件でしか問題が発生せず、検出が難しい場合がある。いわゆる**隠れたバグ**、になることがある。



サニタイザによる検出でブレークが発生し、IDEに問題のあるアクセスの詳細情報が表示されます。





動画で詳しく見る

ダウンロード　かしこいエージェント、ベアメタルローダー

- OS非依存でMMU対応のベアメタルローダーは、ELFのヘッダを参照し、仮想・物理アドレスのMMUマップを判断するかしこいローディング機能をもっています。
- IDEとローダーが連携し、作成しなおしたモジュールだけを再ロードして実行できます。
- ブートアップ、メモリ初期化、ローディングについて、SOLID がフレームワークを提供します。

動作確認　シミュレーション・エミュレーション

- 既存のシミュレータ（QEMU）、エミュレータ（PARTNER-Jet2）に加え、モニタデバッガをオプションで提供します。

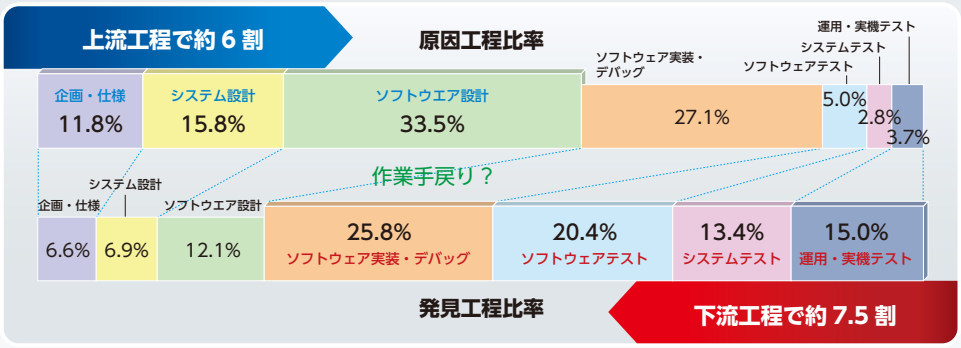
動かす

enjoy Development

SOLIDは開発・検証を計画通りに進めるためのツールです。

プロジェクト管理者の悩み

組込システムの不具合の約6割は
ソフトウエア設計・実装段階で作り込まれており、発見するのはその7割以上がデバッグ・テスト工程というのが実態です。
効率よい開発・検証にはどのようなツールが有効でしょうか？



2012 年版組み込みソフトウェア産業実態把握調査報告書

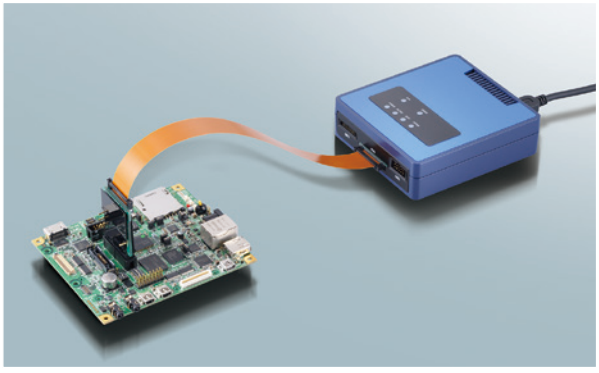
SOLIDプラットフォームのメリット

こんな悩みを・・・

- 最先端のOSSツールを採用したいけれど、社内で使用実績がないとツールを使いこなせるか不安。
- 長いシンボル名や、類似のシンボル名をうっかりタイプミス、こういうミスはレビューでもなかなか見つからない。
- コード修正しながらデバッグしているときは、ダウンロード時間が長いと開発者の集中が途切れてしまう。
- システムテストでは膨大な条件組み合わせを検証しなければいけないし、なにより開発委託先と検証条件をすり合わせるのに苦労する。

SOLIDが解決

- ☑ 操作性に定評のあるVisual Studioベースのユーザーインタフェースを採用しているので、ツール習得に時間がかかりません。
- ☑ OSやミドルウエアをツールと一体化してKMCが提供、技術サポートもするので安心です。
- ☑ コード入力補完機能のあるかしこいエディタを採用、効率のよいタイピングができ、タイプミスも防ぎます。
- ☑ スマートなローダーが、デバッグ中の機能モジュールだけを選択的に短時間でダウンロードします。
- ☑ LLVM/Clangコンパイラの持つ豊富な静的・動的解析ツール機能を利用できるので、従来はシステムテストで稀にしか検出できないようなメモリリソース競合不具合などを、評価作業の初期段階で検出します。



実績豊富なJTAGエミュレータ

PARTNER Jet2

- 各種ARMプロセッサ対応（64bit Cortex-A57 など対応済み）
- USBバスパワー対応（Model 10）
- プローブホットプラグ対応
- マルチコアデバッグ、SMPデバッグ
- ETM HSSTP（LVDSでの高速トレース対応）
- 各種ロギングモード
- QProbeによる動的性能解析
- Linux、GCCコンパイラ、LLVM/Clangコンパイラ対応

KMC
京都マイクロコンピュータ株式会社
本　　社：〒610-1104 京都市西京区大枝中山町2-44　Tel.075-335-1050
東京オフィス：〒105-0004 東京都港区新橋2-14-4 Rビル5F　Tel.03-5157-4530
http://www.kmckk.co.jp/
お問い合わせメールアドレス：jp-info@kmckk.co.jp

2017.5