在 WEC7 主板上启动 VFP 硬件浮点处理器

英创公司

2017 年 3 月

关于 ARM 指令

英创公司开发的基于 WEC7 的工控主板目前包括 3 种型号：

主板型号	CPU 架构	其他重要技术指标
ESM6802	Cortex-A9 双核	ESMARC 主板架构体系，主推产品型号
ESM3354 / ESM3352	Cortex-A8	ESMARC 主板架构体系，主推产品型号
EM335x / EM3352	Cortex-A8	成熟产品型号

在使用英创的 WEC7 主板时，用户需要使用 Visual Studio 2008（简称 VS2008）来开发其应用程序。尽管 Cortex-A8 和 Cortex-A9 处理器均支持性能更高的 ARMv7 指令集，但微软在 VS2008 中所仍然使用 ARMv4i 指令集的通用 arm 编译器（编译器版本号为：15.00.20720）。而 A8、A9 处理器所带的矢量浮点处理器（Vector Float-Point Processor）都需要在 ARMv7 指令下才能正常启动运行。换句话说，在 ARMv4i 指令集下，对浮点的处理仍然是采用软件仿真包来实现，而没有用到高端 ARM 处理器自带的硬件浮点处理器。这对涉及大量浮点处理应用的客户来说是很遗憾的事。

本文将以 ESM3354 为测试平台，介绍在现有 VS2008 基础上实现硬件浮点处理的方法。

编译器及 SDK 的准备

我们为需要浮点处理的客户准备了 ARMv7 编译工具以及基于 ARMv7 工具的 SDK，具体如下表所示：

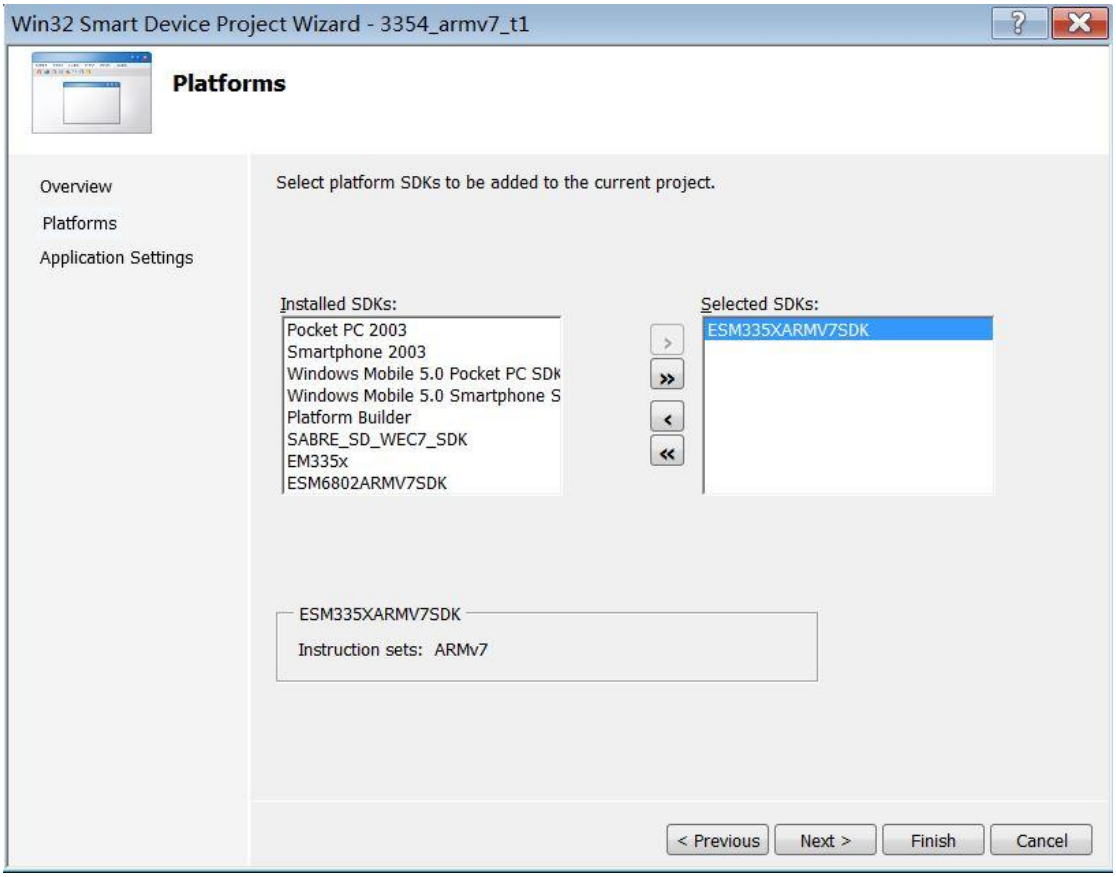
ARMv7 编译工具包	ms-armv7-compiler.tar 编译器版本 15.01.50304.03
ESM335x SDK	ESM335XARMV7SDK.msi
ESM6802 SDK	ESM6802ARMV7SDK.msi

客户需要首先安装新的 AMRv7 的 SDK，ARMv7 的 SDK 与原来的 ARMv4i 的 SDK 是独立并行的，并不需要卸载原来的 SDK。安装完成后，再把 ARMv7 编译器工具包解压到本地硬

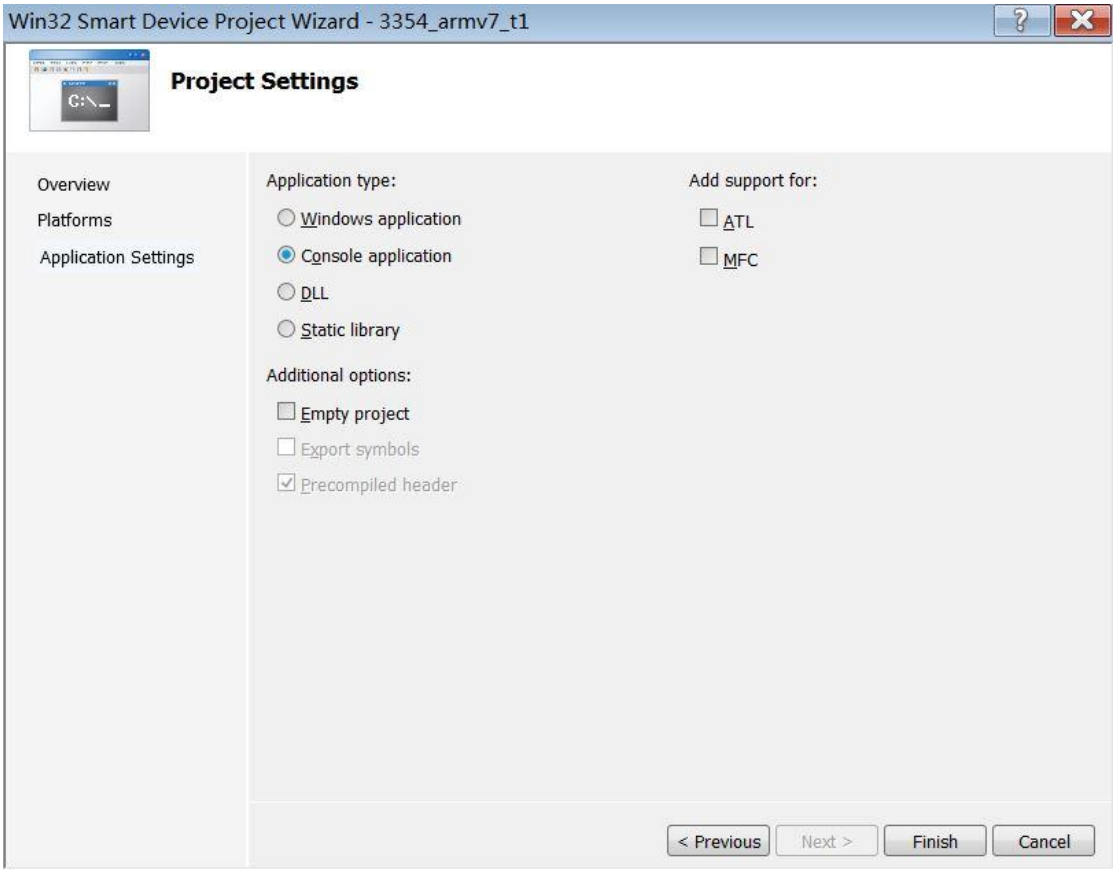
盘上，例如 D:\ms-armv7-compiler。

建立测试程序 1

打开 VS2008，建立测试程序 3354_armv7_t1，平台（platform）选择 ESM335XARMV7SDK，用户可以看到其指令集显示为 armv7：



点击<next>，选择 Console Application，



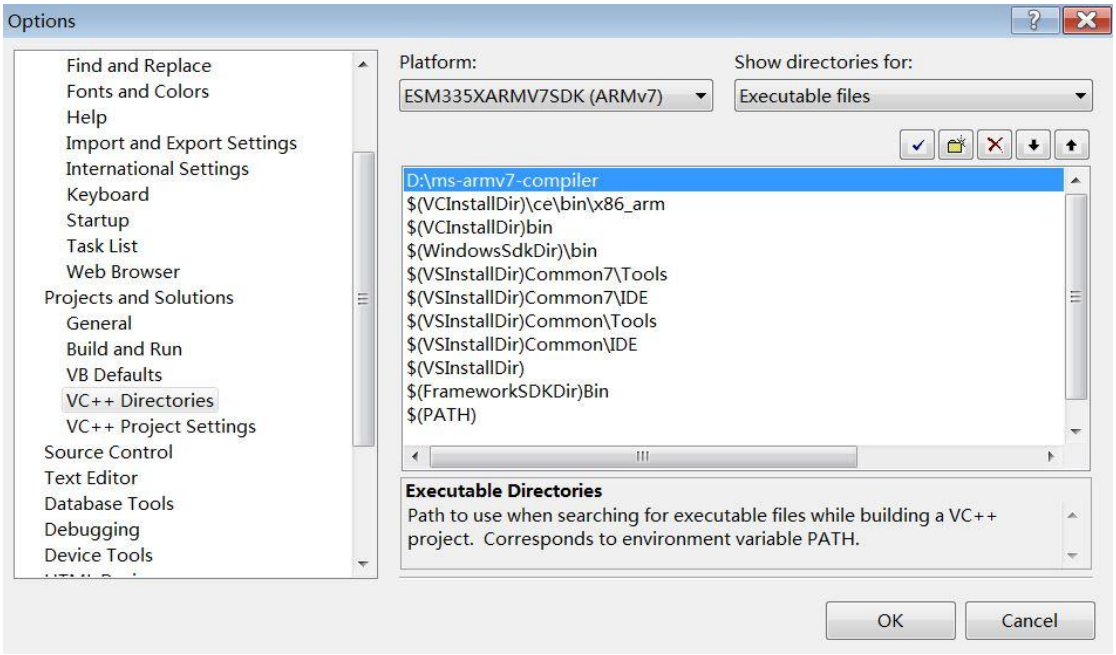
最后点击<finish>，进入代码窗口。以下是测试的完整代码，客户可拷贝粘贴到所生成的代码窗口区域中。

```
// 3354_armv7_t1.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"

int _tmain(int argc, _TCHAR* argv[])
{
    double  f1 = 2.200002;
    double  f2 = 2.200001;
    double  ans = 1.0;
    longiterations = 5 * 1000 * 1000;
    DWORD   dwStartTick, dwEndTick;

    _tprintf(TEXT("Microsoft compiler version: %d\r\n"),_MSC_FULL_VER);
    _tprintf(TEXT("ARM instruction set: %d\r\n"),_M_ARM);

    if(argc > 1)
    {
        f1 = _wtof(argv[1]);
    }
```

```
    if(argc > 2)
    {
        f2 = _wtof(argv[2]);
    }

    if(argc > 3)
    {
        iterations = _wtoi(argv[3]) * 1000 * 1000;
    }

    dwStartTick = GetTickCount();
    wprintf(L"f1=%f f2=%f loop=%d starting...%d\r\n", f1, f2, iterations,
dwStartTick);

    for(int i=0; i<iterations; i++)
    {
      ans *= f1;
      ans /= f2;
    }

    dwEndTick = GetTickCount();
    wprintf(L"ans = %f %d loop/msec end...%d\r\n", ans,
(int)(iterations/(dwEndTick - dwStartTick)), dwEndTick);
    return 0;
}
```

设置编译器路径

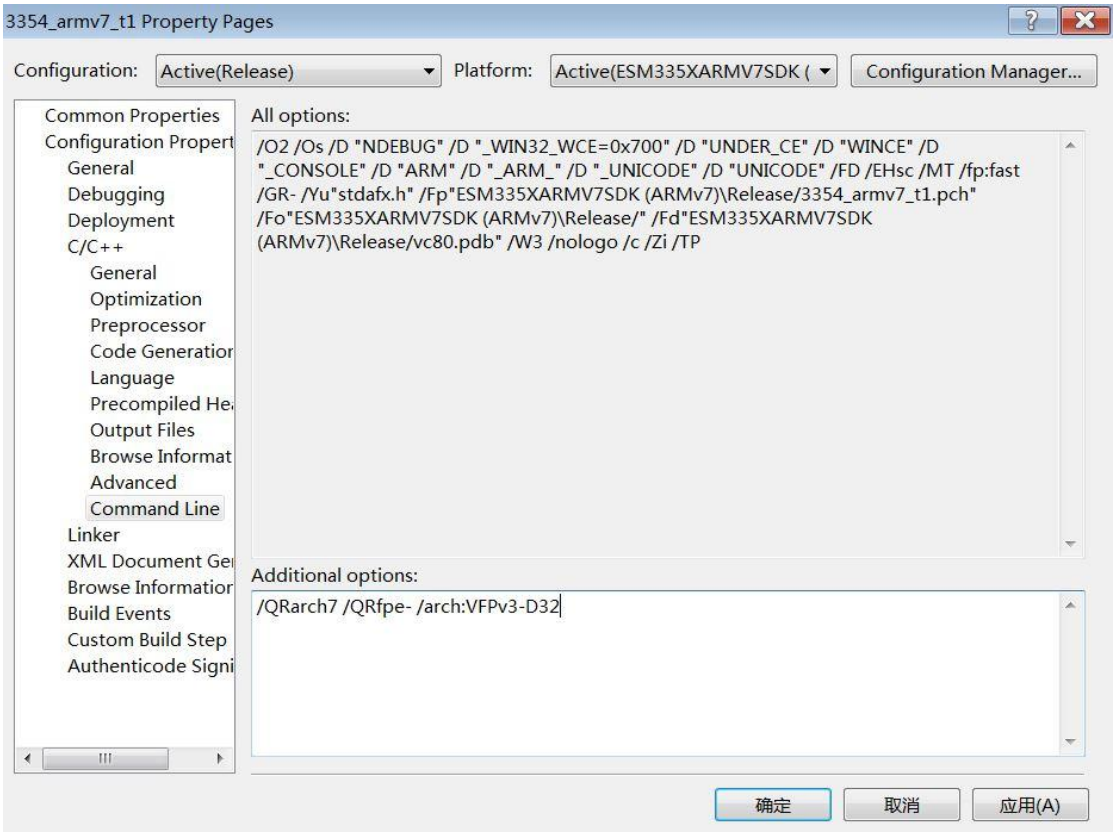
准备好测试程序后，首先需要把 ARMv7 编译器加入到 VS2008 中，具体设置方法是在菜单栏中选择 Tools -> Options -> Projects and Solutions -> VC++ Directories，然后选择页面的 Platform 栏目中选择 ESM335XARMV7SDK(ARMv7)，最后在路径栏目中添加新路径如下：



编译器路径设置，对一个平台只需要设置一次。也就是对本机说，当创建其他基于 ESM335XARMV7SDK 的应用程序时，都不需要再重复设置编译器路径了。

设置应用程序编译链接选项

从工具栏点击 Project -> Properties -> C/C++ -> Command Lines，添加 ARMv7 指令选项以及浮点处理选项：“/QRarch7 /QRfpe- /arch:VFPv3-D32”如下：



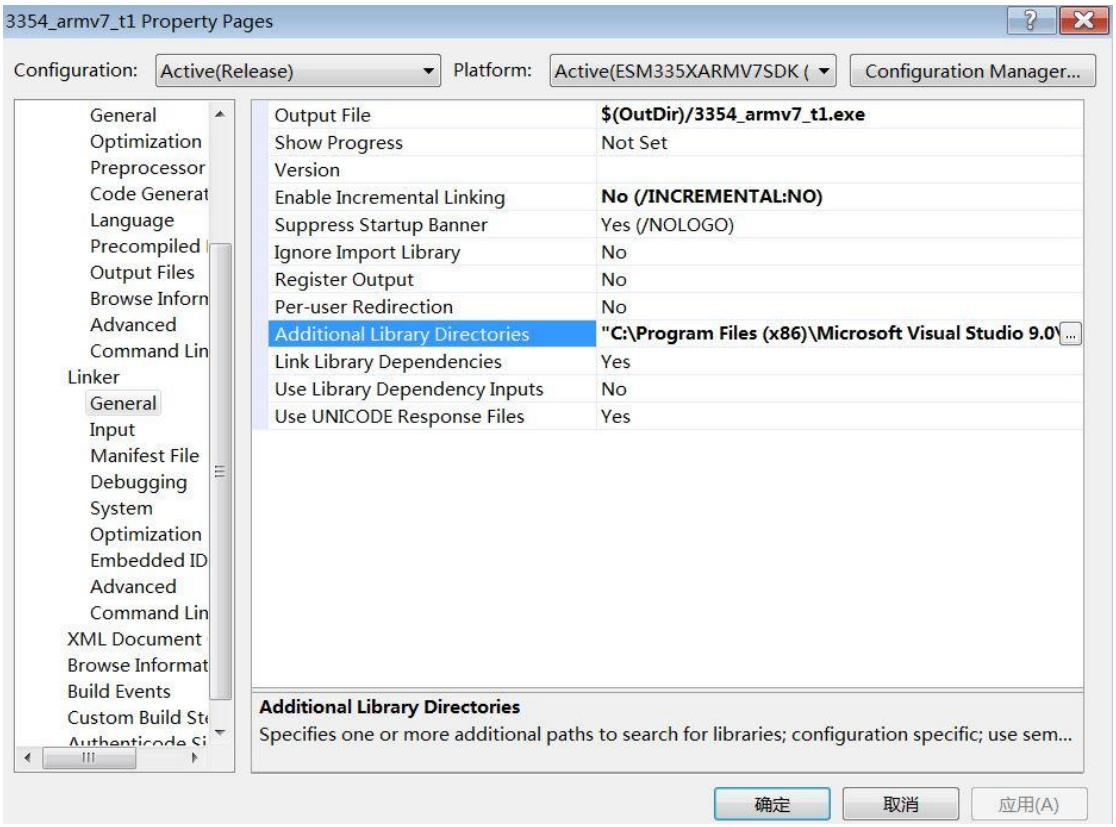
各个选项的功能微软的大致解释如下：

- /QRarch7 // -> ARMv7 Architecture
- /QRfpe- // -> Enable Hardware Floating-Point Targeting (Compact 7)
- /arch:VFPv3-D32 // -> 使用 VFP 内部的 32 个 64 位寄存器

由于 ARMv7 编译器并不知道 VS2008 的基本环境库路径，也需要通过菜单添加：Project -> Properties -> Linker -> General，在 Additional Library Directories 栏加入：

“C:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\ce\lib\armv4i”

设置的界面为：



设置完成后，就可 Build 测试程序，并下载至目标板上运行。该测试程序在 ESM335x 和 EM335x 两个系列的主板上均可正常运行：

```
\nandflash> 3354_armv7_t1
Microsoft compiler version: 150150304
ARM instruction set: 7
f1=2.200002 f2=2.200001 loop=5000000 starting...20484726
ans = 9.705820 19455 loop/msec end...20484983
```

这里“19455 loop/msec”表示每 ms 的循环次数，是反应计算能力的主要参数。

测试结果比较

按构造上述测试程序的方法，可方便地构造一个使用 VS2008 缺省编译器（ARMv4i）的 t1 程序，以进行比较。进一步地，我们在 t1 主循环中增加一行超越函数计算，构成测试程序 t2：

```
for(int i=0; i<iterations; i++)
{
  ans *= f1;
  ans /= f2;
  ans = cos(ans);       //t2新加代码
}
```

为了更全面比较测试结果，我们还在 Linux 版本上运行相同测试程序，进行编译器的横向对比。

整个测试的综合结果如下表所示：

	t1	t2
CL-15.00.20720 (armv4i)	6858 loop/ms	644 loop/ms
CL-15.01.50304.03 (armv7)	19455 loop/ms	687 loop/ms
Improvement	2.83 (+183%)	1.07(+7%)
Gcc-4.4.1+soft (armv7-a)	1921 loop/ms	470 loop/ms
Gcc-4.4.1+vfp (armv7-a)	13857 loop/ms	587 loop/ms
improvement	7.2 (+620%)	1.25 (+25%)

上表的改进栏目中是 ARMv7 运行速度相对 ARMv4i 速度的倍数，括号内为提高的百分比。测试程序 t1 评估的是常规的浮点计算，使能 VFP 后，是有明显改进的；而且 Linux 版本的相对改进更大。但加入超越函数后，VFP 的硬件优势就几乎全部丧失了。我们理解 ARM 的 VFP 与传统 x86 的协处理器（Co-processor）在对超越函数的处理上还是差别很大的，基本上还是采用多项式级数来合成的，本来设置 VFP 的主要目的也是面向数字滤波、图形处理这类以乘加为主要特色的浮点处理，对数学超越函数的处理确实不在行。

小结

对需要大量使用乘加类型浮点处理的应用，采用本文的方法启动 ARMv7 指令及 VFP 浮点处理器，是能够大大改善应用程序的性能的。对超越函数的处理，需要转换成表格方式处理，而规避直接的计算，以保持程序总的处理性能。

微软在 VS2008 配置的 ARMv4i 编译器，其性能明显优于 Linux 平台中使用的开源 GCC 编译器，估计这也是微软保留这款编译器的原因之一。关于 ARMv7 指令相对 ARMV4i 指令的性能比较，第三方公司已做详细测试，感兴趣的客户可向英创索取相关文档。

有需求客户可向英创索取 ARMv7 编译器以及相应的 SDK，英创公司的技术支持邮箱为：support@emtronix.com 。