CodeWarrior 10. Инструменты для профилирования и анализа работы контроллеров Freescale.

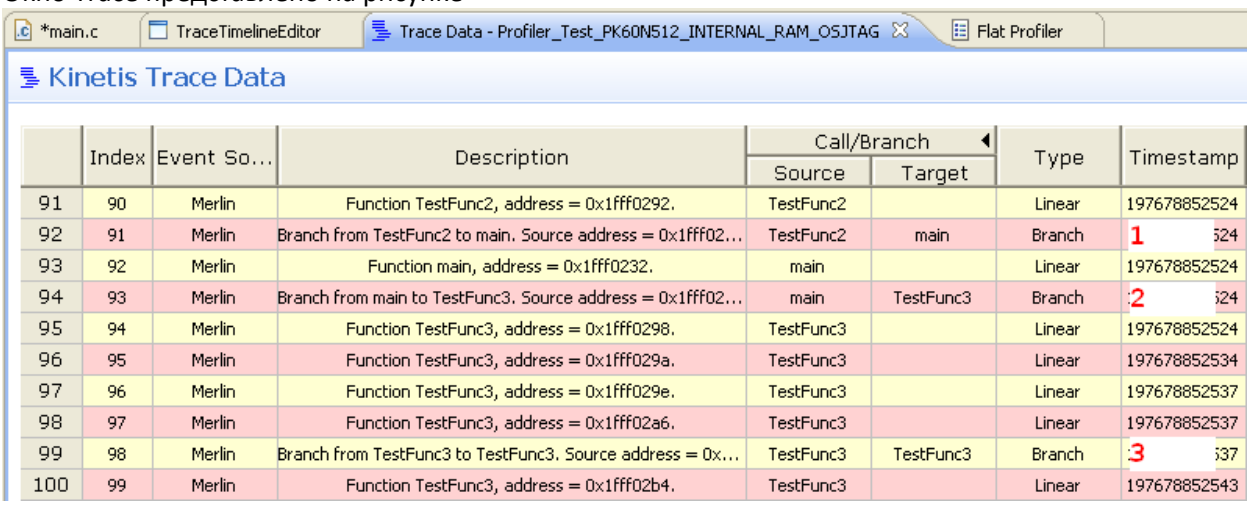
IDE CodeWarrior содержит средства для анализа и профилирования программного обеспечения. Встроенный профилировщик поддерживает работу с контроллерами семейств HCS08, ColdFire V1-V4, Kinetis.

Основные доступные инструменты:
- Trace;
- Time Line Editor;
- Flat Profiler;

В качестве примера будем использовать простую программу, состоящую из вызовов нескольких функций и выполняющуюся на контроллере **Kinetis K60**:

```
void main(void)
{
    int counter = 0;

    TestFunc1();
    TestFunc2(10);
    TestFunc3(0);
    TestFunc2(5);
    TestFunc3(2);

    while(1)
            counter++;
}

void TestFunc1(void) {     // Simple function
    volatile int i;
    volatile int n;
    volatile int k;
    i = 5; n = 10; k = i*n;
    k--; i-= k; n-= i;
}

void TestFunc2(int n) {    // Function with loop
    volatile int k = n;
    while(k--);
}

void TestFunc3(int n)      { // Function with branch
    if(n % 2)
            asm {nop; nop;nop;
            }
    else
            asm {nop;nop;nop;nop;nop;nop;nop;nop;nop;
            }
}
```

Trace.
Окно Trace представлено на рисунке



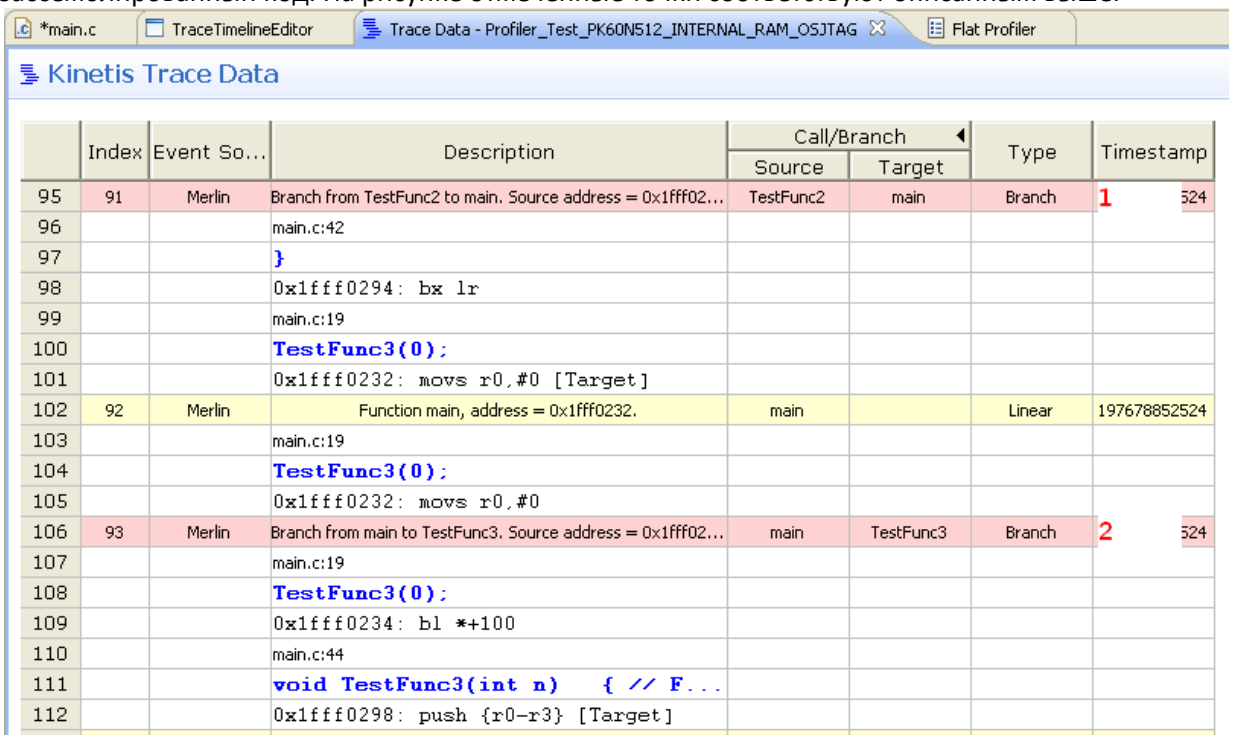
	Index	Event So...	Description	Call/Branch Source	Call/Branch Target	Type	Timestamp
91	90	Merlin	Function TestFunc2, address = 0x1fff0292.	TestFunc2		Linear	197678852524
92	91	Merlin	Branch from TestFunc2 to main. Source address = 0x1fff02...	TestFunc2	main	Branch	1 524
93	92	Merlin	Function main, address = 0x1fff0232.	main		Linear	197678852524
94	93	Merlin	Branch from main to TestFunc3. Source address = 0x1fff02...	main	TestFunc3	Branch	2 524
95	94	Merlin	Function TestFunc3, address = 0x1fff0298.	TestFunc3		Linear	197678852524
96	95	Merlin	Function TestFunc3, address = 0x1fff029a.	TestFunc3		Linear	197678852534
97	96	Merlin	Function TestFunc3, address = 0x1fff029e.	TestFunc3		Linear	197678852537
98	97	Merlin	Function TestFunc3, address = 0x1fff02a6.	TestFunc3		Linear	197678852537
99	98	Merlin	Branch from TestFunc3 to TestFunc3. Source address = 0x...	TestFunc3	TestFunc3	Branch	3 537
100	99	Merlin	Function TestFunc3, address = 0x1fff02b4.	TestFunc3		Linear	197678852543

В окне мы можем проследить последовательность выполнения функций и переходов. В поле «Description» приводится описание для каждой отладочной записи.

Точка 1 – возврат из функции **TestFunc2()** в **main()**.

Точка 2 – переход на функцию **TestFunc3()** и соответственно предыдущая строка это вызов функции.
Точка 3 – выполнение условного перехода внутри функции **TestFunc3()**.

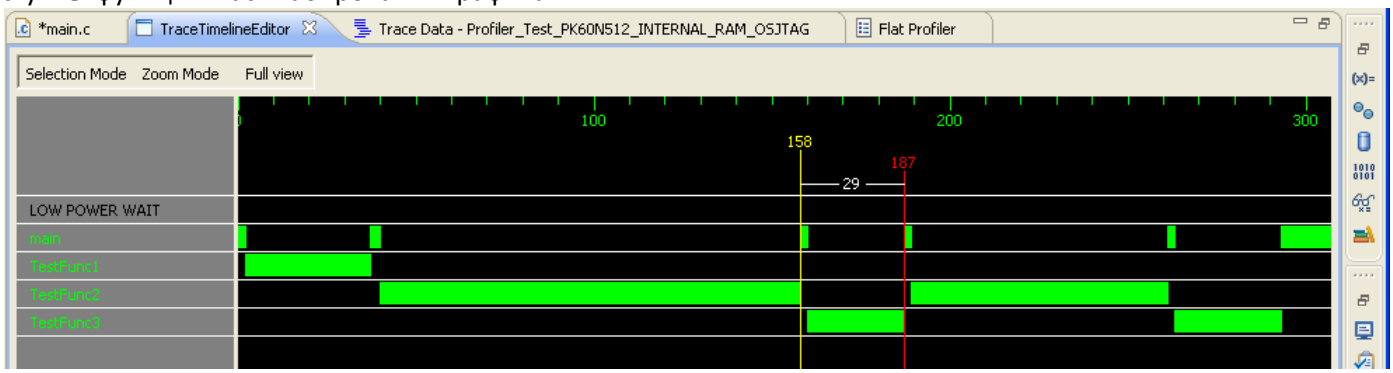
Развернув описание, можно увидеть подробную информацию по каждой записи, включая С-код и дизассемблированный код. На рисунке отмеченные точки соответствуют описанным выше.



	Index	Event So...	Description	Call/Branch Source	Call/Branch Target	Type	Timestamp
95	91	Merlin	Branch from TestFunc2 to main. Source address = 0x1fff02...	TestFunc2	main	Branch	1 524
96			main.c:42				
97			}				
98			0x1fff0294: bx lr				
99			main.c:19				
100			TestFunc3(0);				
101			0x1fff0232: movs r0,#0 [Target]				
102	92	Merlin	Function main, address = 0x1fff0232.	main		Linear	197678852524
103			main.c:19				
104			TestFunc3(0);				
105			0x1fff0232: movs r0,#0				
106	93	Merlin	Branch from main to TestFunc3. Source address = 0x1fff02...	main	TestFunc3	Branch	2 524
107			main.c:19				
108			TestFunc3(0);				
109			0x1fff0234: bl *+100				
110			main.c:44				
111			void TestFunc3(int n) { // F...				
112			0x1fff0298: push {r0-r3} [Target]				

Time Line Editor.

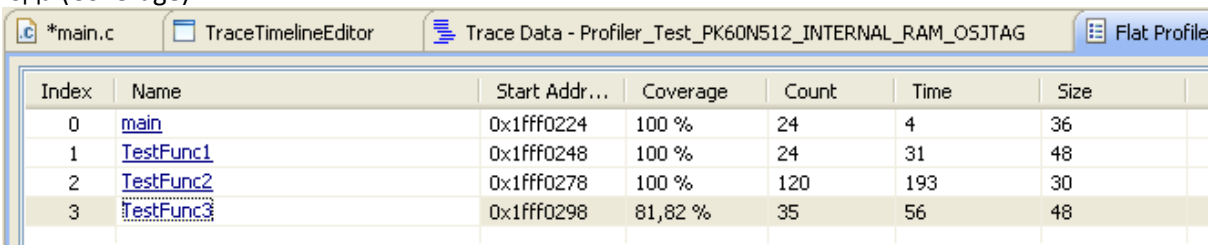
Time Line Editor предоставляет наглядную диаграмму вызовов функций их продолжительность и соотношение периодов работы, позволяет оценить время выполнения каждого участка программы или целой группы. Доступны функции масштабирования графика.



По диаграмме видно вызовы функций **TestFunc** и их длительность. Для первого вызова **TestFunc3()** показано время выполнения с учетом вызова функции из **main()**.

Flat Profiler.

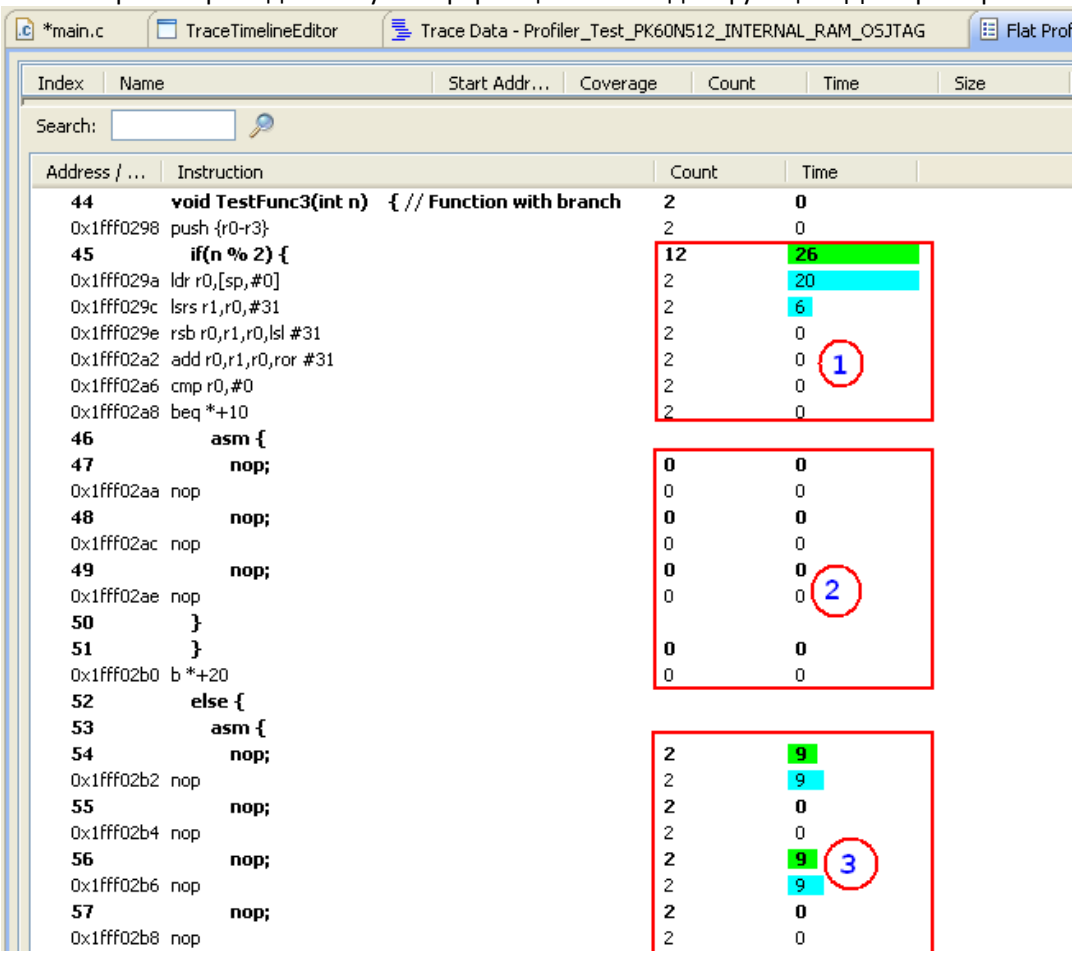
Данный сервис предоставляет полную информацию по работе функций. Считается кол-во выполненных инструкций, время затраченное на каждую функцию и инструкцию, общий размер функций, выполняется анализ покрытия кода (Coverage).



Index	Name	Start Addr...	Coverage	Count	Time	Size
0	main	0x1fff0224	100 %	24	4	36
1	TestFunc1	0x1fff0248	100 %	24	31	48
2	TestFunc2	0x1fff0278	100 %	120	193	30
3	TestFunc3	0x1fff0298	81,82 %	35	56	48

Разберем функцию **TestFunc3()**. Мы можем видеть адрес входа в функцию 0x1FFF0298, покрытие кода (процент инструкций, которые выполнялись внутри функции от общего количества инструкций) – 81,82%, количество реально выполненных инструкций – 35, время в клоках, которое потребовалось для выполнения функции и общий размер функции – 48 байт.

Можно просмотреть детальную информацию по каждой функции. Для примера возьмем также **TestFunc3().**



В процессе работы эта функция вызывалась два раза. При этом оба раза был указан четный параметр, поэтому в обоих случаях выполнялась одна и та же ветвь. Можно заметить в сегментах 1 и 3, что каждая команда функции выполнялась 2 раза и ни одной команды не было выполнено из сегмента 2. Напротив каждой команды указано суммарное количество тиков системной частоты на данную команду. С учетом, что сегмент 2 не выполнился ни разу, анализатор сообщил, нам, что только 81,82% кода функции реально выполнялось.

В целом, профилировщик CodeWarrior имеет мощные и удобные средства для оптимизации программного обеспечения. Однако, следует иметь в виду, что поток информации от ядерных средств отладки очень большой. По заполнению встроенного буфера отладочных сообщений, процессор полностью блокируется на время передачи данных из буфера в отладочную среду. Встроенный в TOWER программатор-отладчик OS-JTAG, существенно замедляет работу контроллера. Поэтому для профилирования рекомендуется использовать более быстродействующие средства, как J-Link, U-Link, J-Link-PRO, J-Trace, U-Link-PRO. Последние имеют скорость работы достаточную для отладки ПО в реальном времени.

Полное руководство по средствам профилирования и отладки можно скачать с сайта Freescale - http://cache.freescale.com/files/soft_dev_tools/doc/user_guide/Profiling_and_Analysis_Users_Guide.pdf?fsrch=1&sr=1