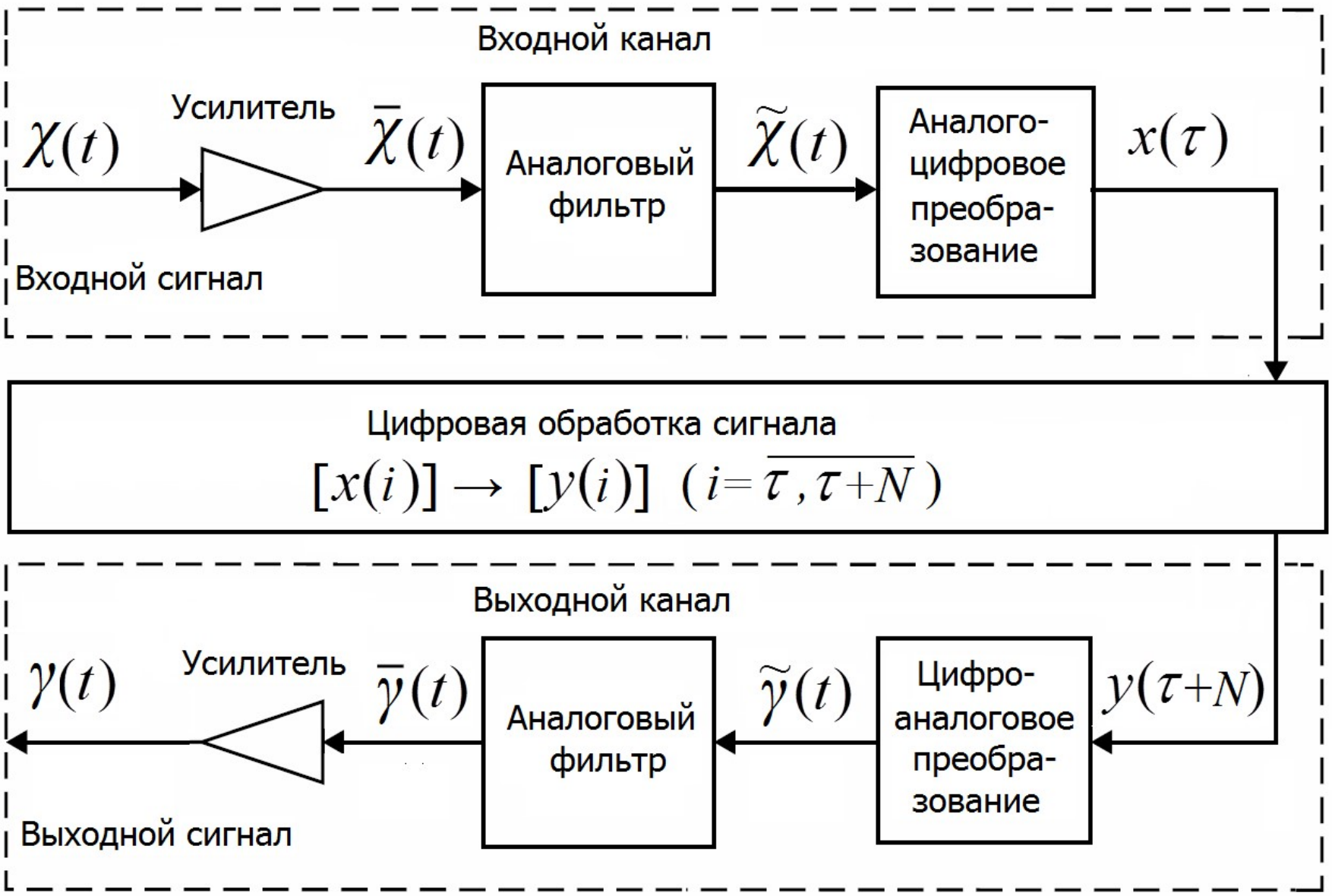
Микропроцессорные устройства обработки сигналов

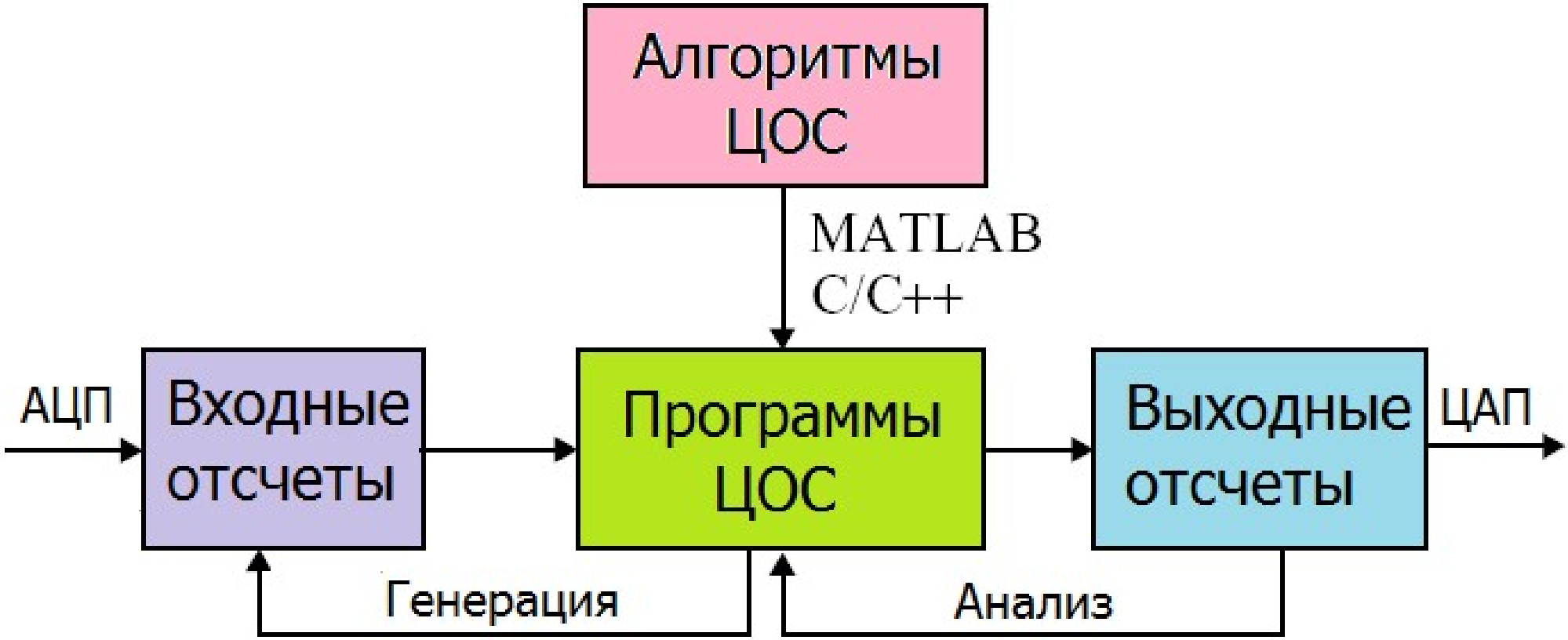
Лекция L18
«Разработка программ»

http://vykhovanets.ru/course67/

Обработка сигналов



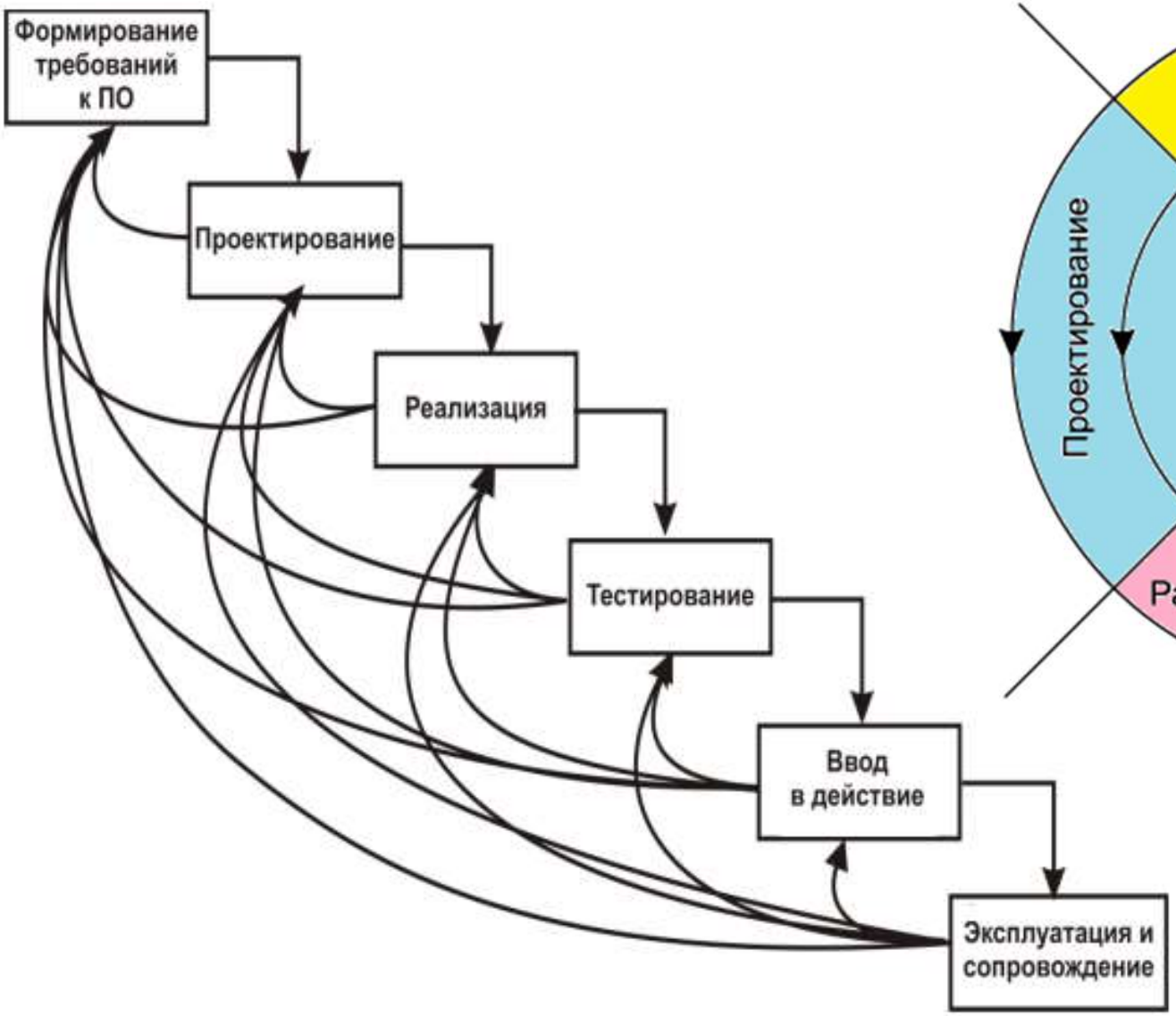
Программа ЦОС

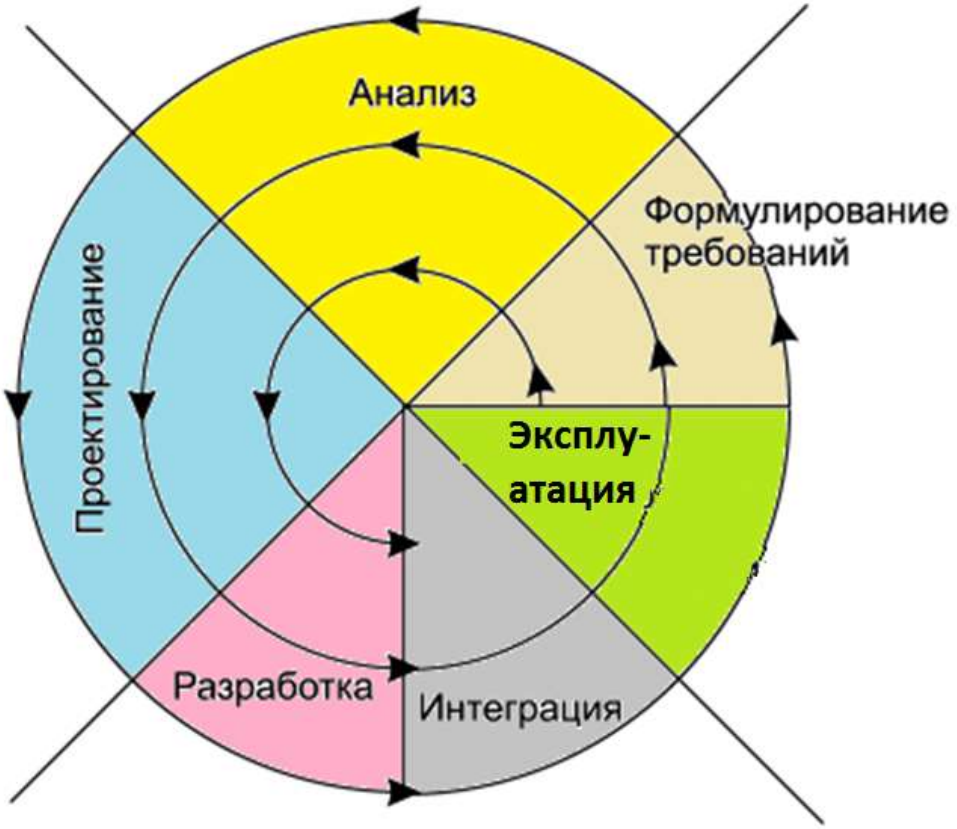


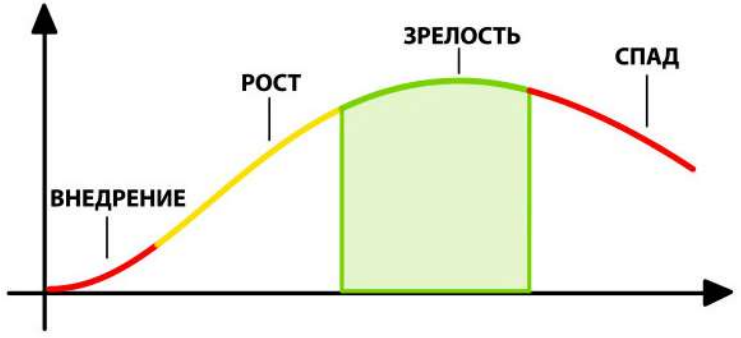
Стадии разработки

ГОСТ 19.102-77

Жизненный цикл

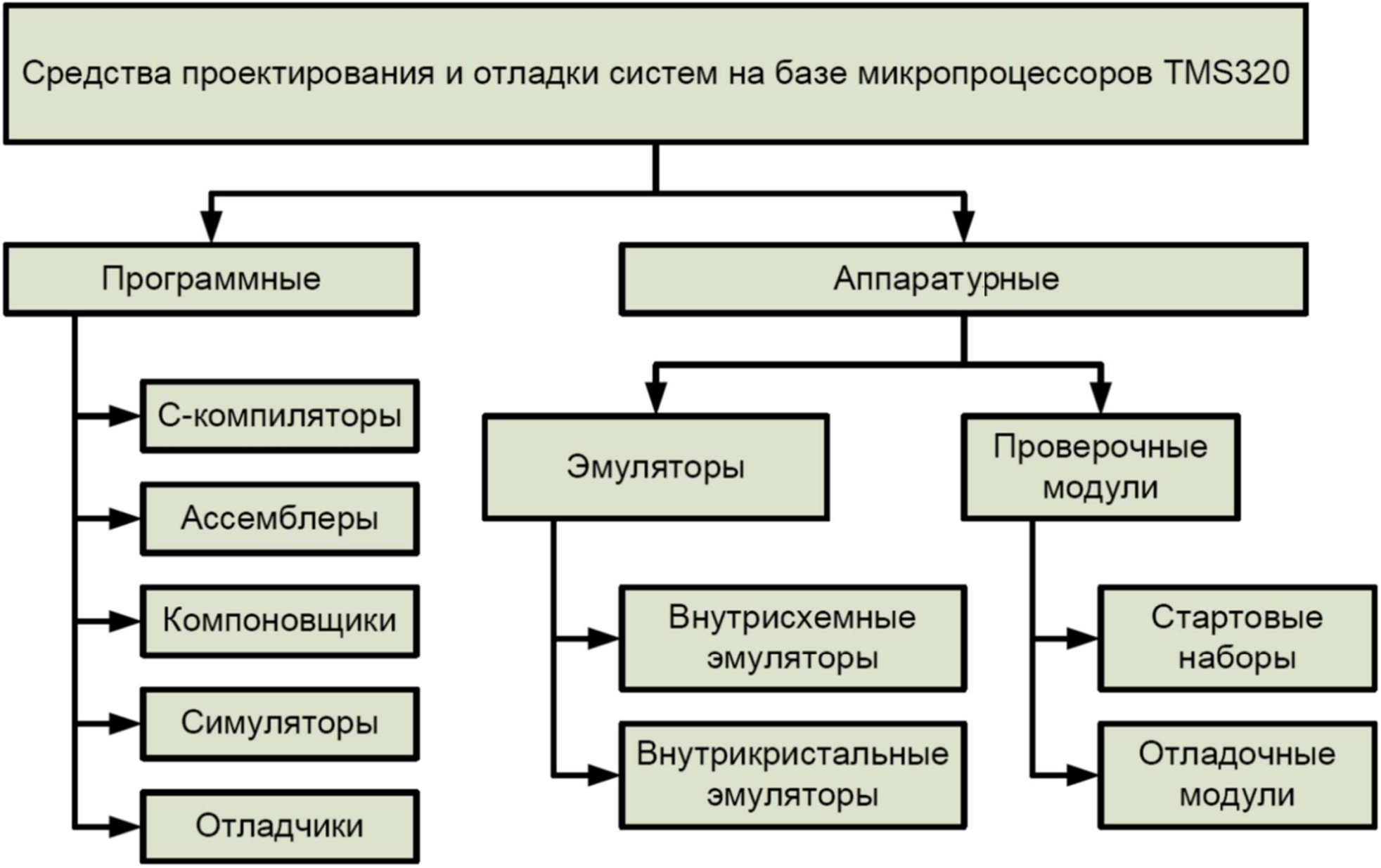




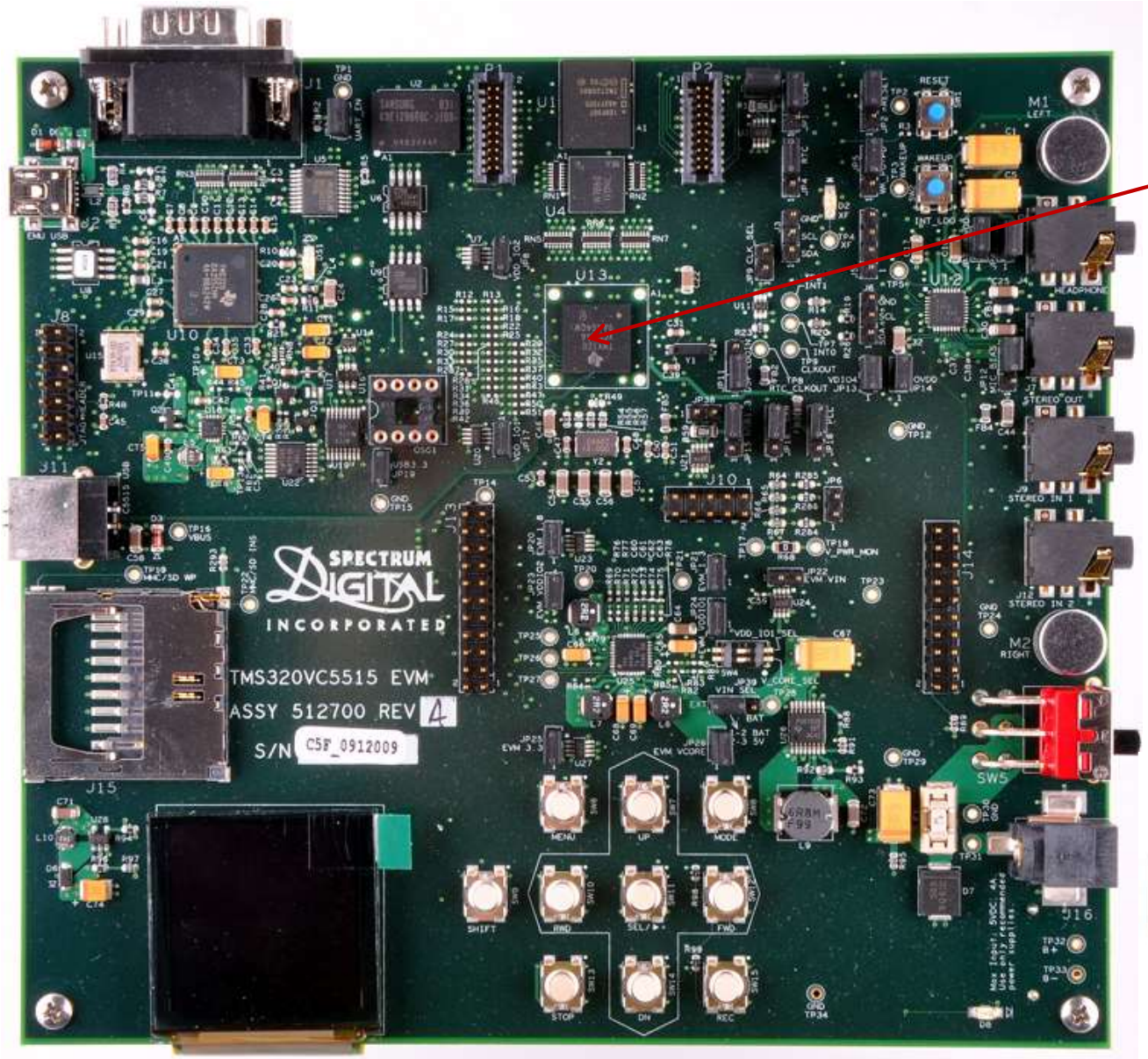


ГОСТ 34.601-90
ISO 12207:1995

Средства разработки

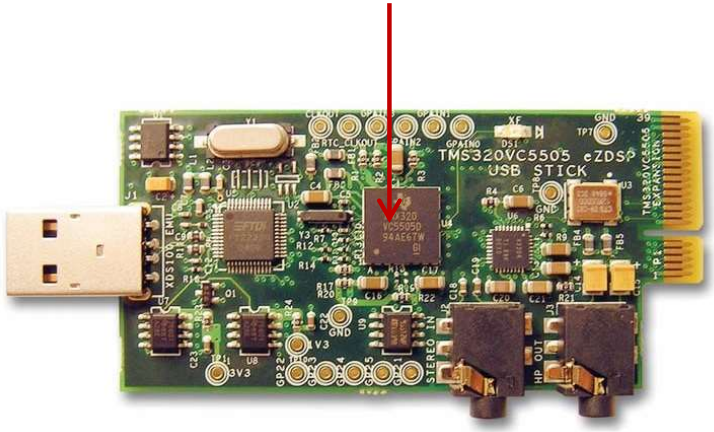


Проверочные модули

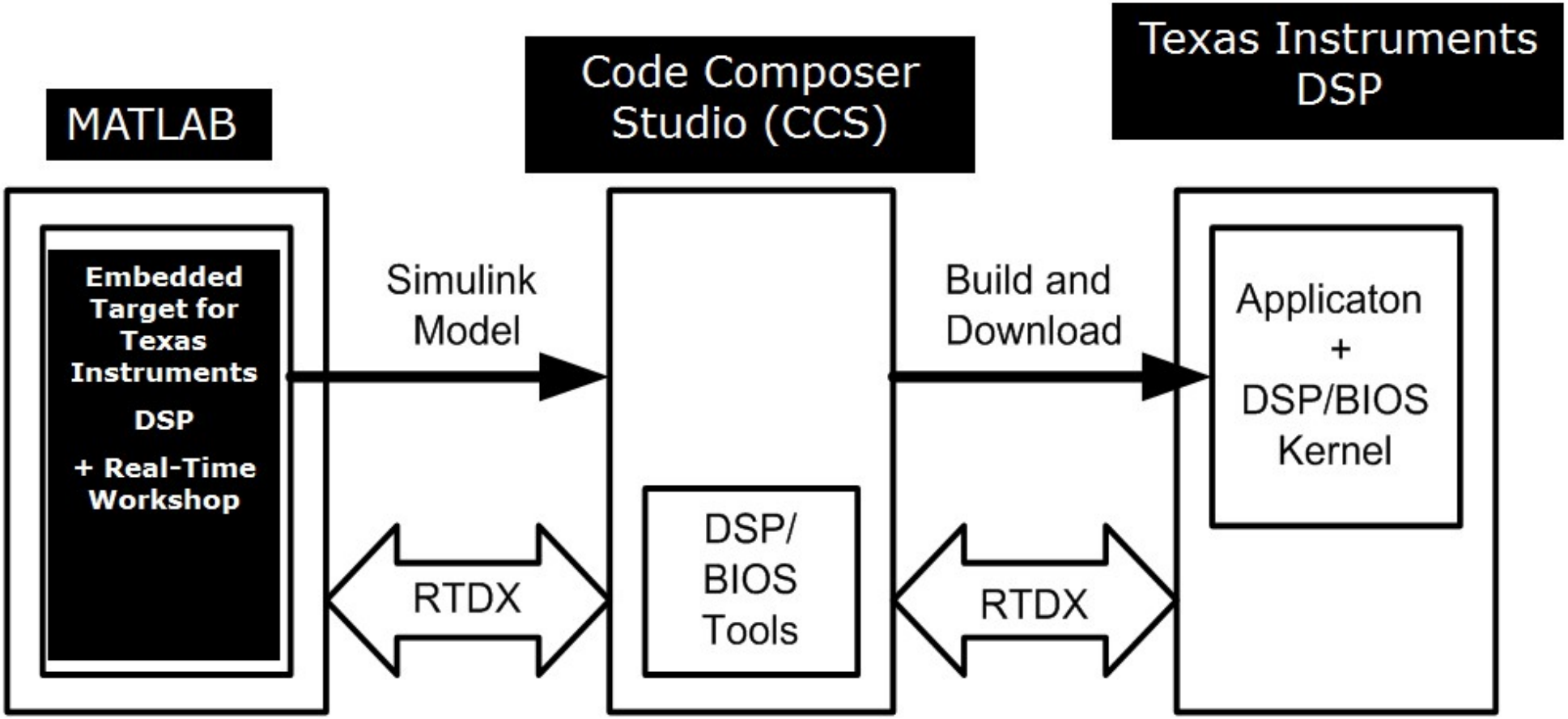


TMS320C5515 EVM
Внутрикристальный эмулятор

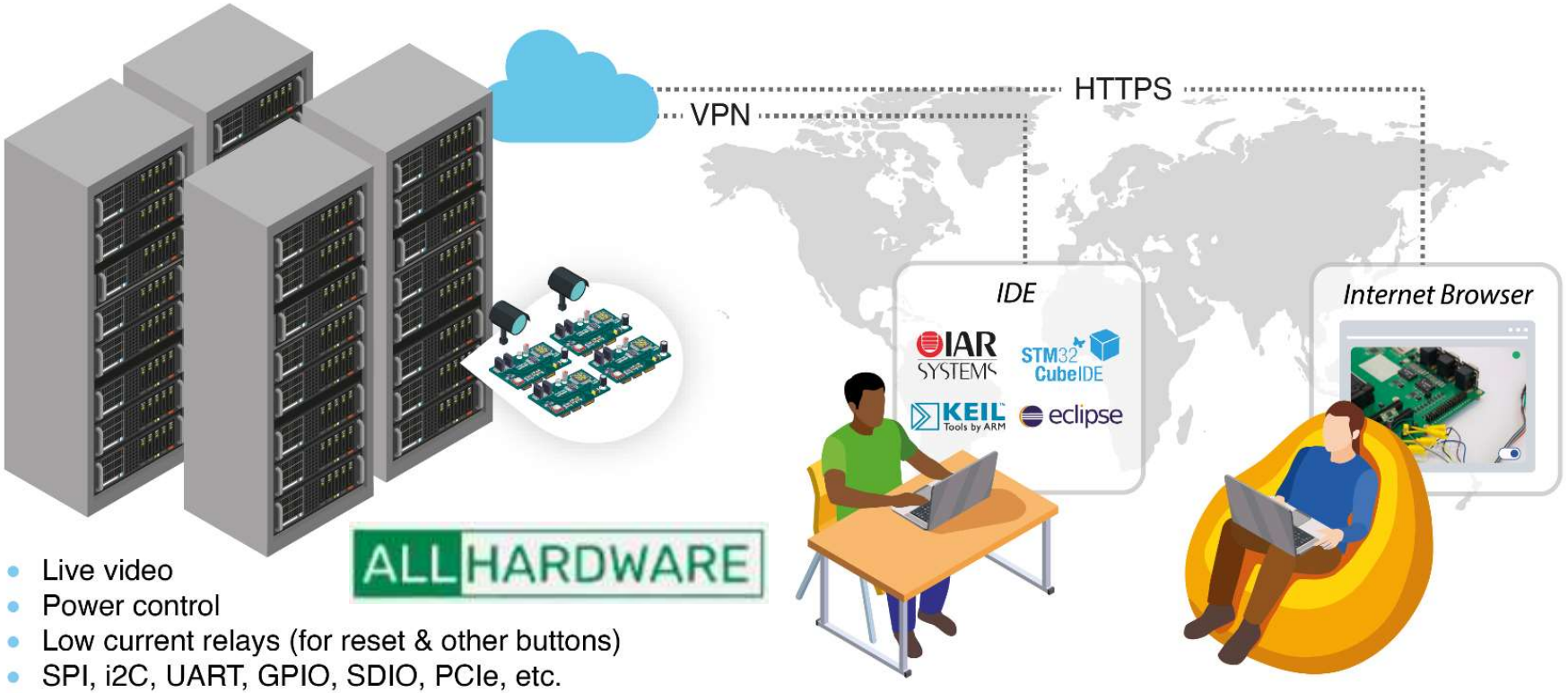
C5505 eZDSP USB Stick
Внутрисхемный эмулятор



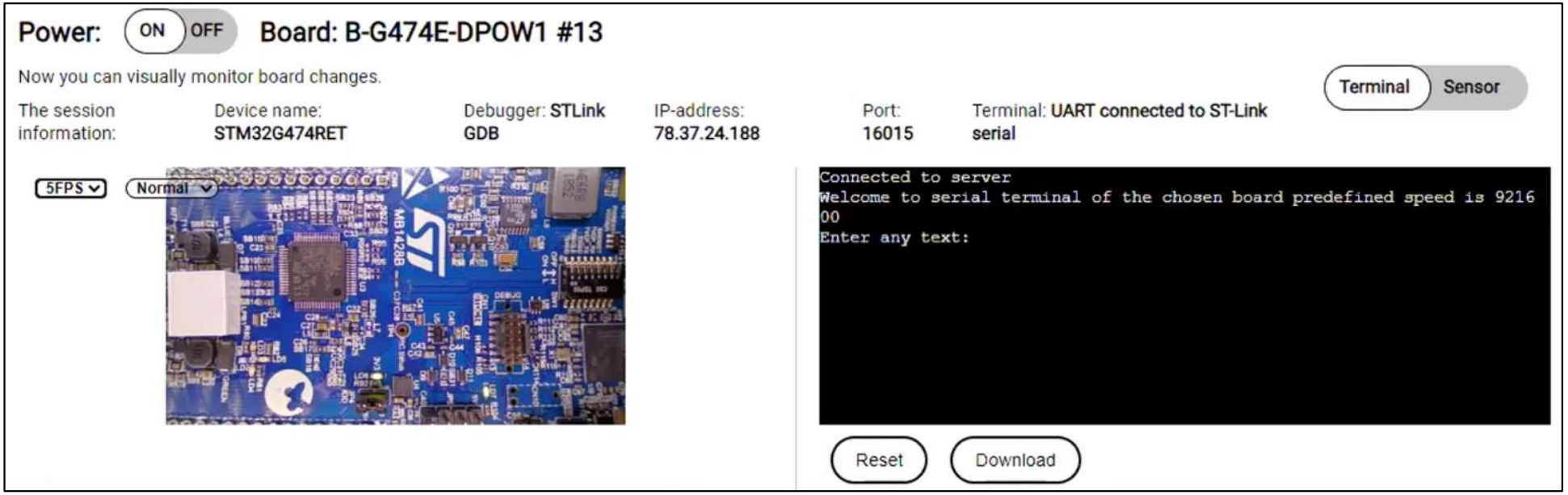
Разработка алгоритмов



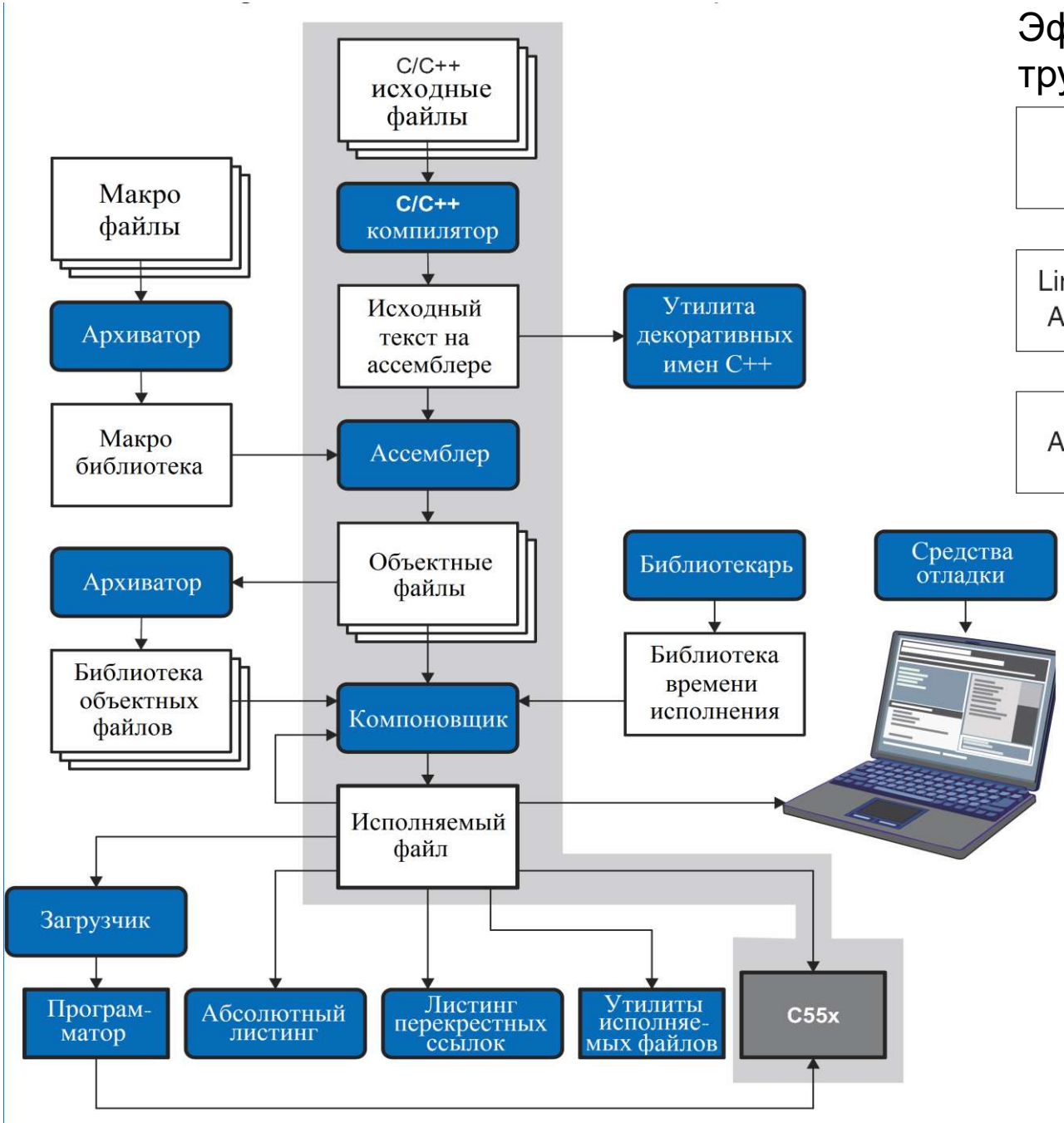
Удаленная отладка



- Live video
- Power control
- Low current relays (for reset & other buttons)
- SPI, i2C, UART, GPIO, SDIO, PCIe, etc.



Разработка программ



Эффективность программ и трудоемкость разработки

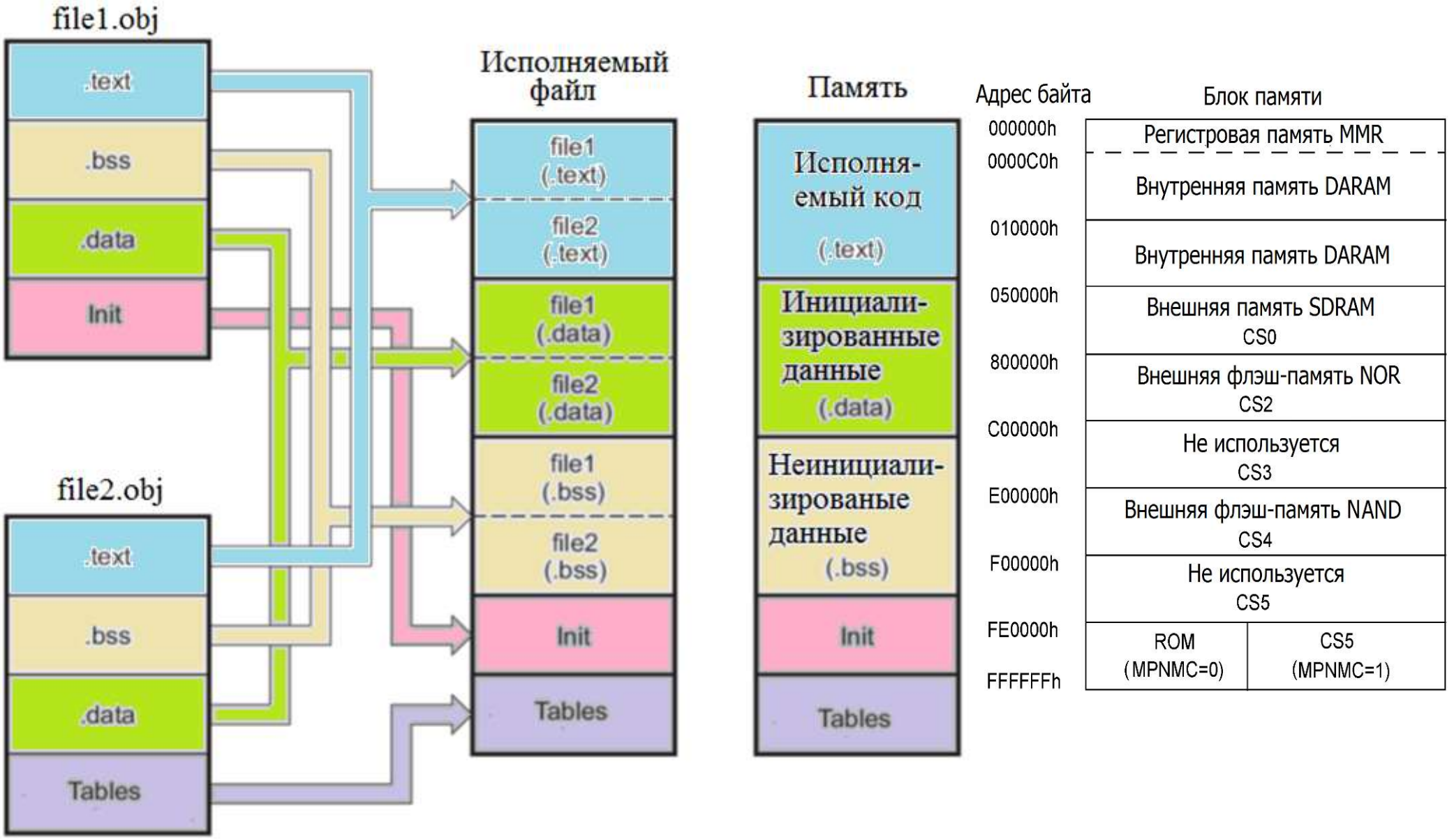
C	Compiler Optimizer	50-80%	Low
Linear ASM	Assembly Optimizer	80-100%	Med
ASM	Hand Optimized	100%	High

Декоративные имена

- Demangling (искажение) C++ names
- Average::insertValue(double, short*, int&) → insertValue_Average**F**dPsRi
- Average::getCount() → getCount_Average**F**v

- d – double, v – void, c – char, f – float, i – int, l – long, s – short, Px – указатель x, Rx – ссылка x, C – const, S – signed, U – unsigned, V – volatile, S – static, An_x – массив из n элементов X, ct – конструктор, dt – деструктор, __pl – оператор +, …

Компоновка модулей



Описание компоновки

```
/* Specify the system memory map */
MEMORY
{
    RAM  (RWIX) : o = 0x000100, l = 0x00feff  /* Data memory     */
    RAM0 (RWIX) : o = 0x010000, l = 0x008000  /* Data memory     */
    RAM1 (RWIX) : o = 0x018000, l = 0x008000  /* Data memory     */
    RAM2 (RWIX) : o = 0x040100, l = 0x040000  /* Program memory  */
    ROM  (RIX)  : o = 0x020100, l = 0x020000  /* Program memory  */
    VECS (RIX)  : o = 0xffff00, l = 0x000100  /* Reset vector    */
}
/* Specify the sections allocation into memory */
SECTIONS
{
    vectors  > VECS   /* Interrupt vector table     */
   .text     > ROM    /* Code                       */
   .switch   > RAM    /* Switch table info          */
   .const    > RAM    /* Constant data              */
   .cinit    > RAM2   /* Initialization tables      */
   .data     > RAM    /* Initialized data           */
   .bss      > RAM    /* Global & static vars       */
   .stack    > RAM    /* Primary system stack       */
   .sysstack > RAM    /* Secondary system stack     */
    expdata0 > RAM0   /* Global & static vars       */
    expdata1 > RAM1   /* Global & static vars       */
}
```

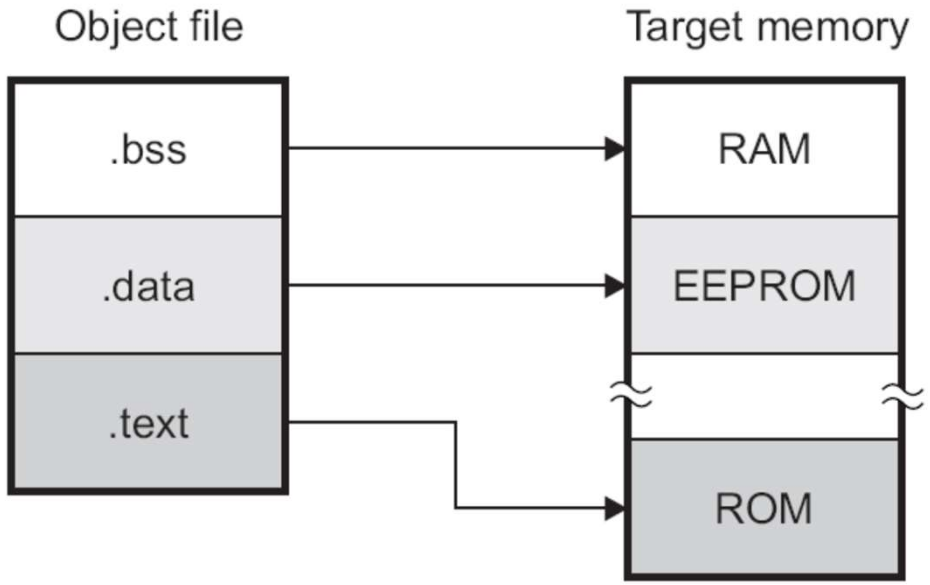
R – чтение
W – запись
X – выполнение
I – инициализация

Секционирование на C

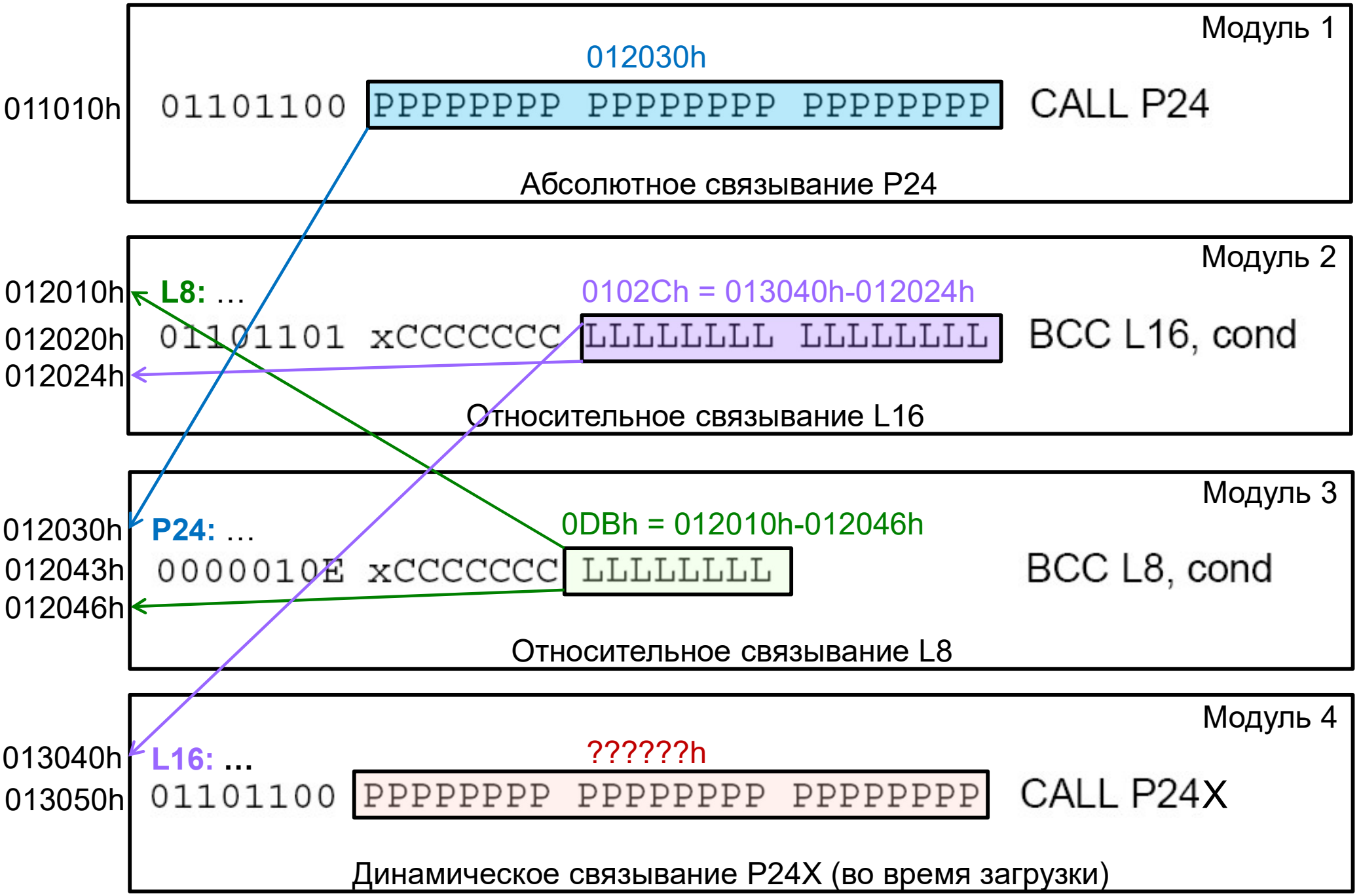
- int x = 0; /* x → .bss, 0 → .cinit */
- char* y = “ab”; /* y → .bss, “ab” → .const */
- char z[ ] = “defgh”; /* z → .data */
- int func(int, int, int);
- void main(void)
  {
  y = func(1,2,3) /* code → .text */
  } /* 1, 2, 3 → .stack */
- int func(int a, int b, int c)
  { /* .stack → a, b, c */
  return a*b+c; /* code → .text */
  }

Размещение секций

Section	Type of Memory	Section	Type of Memory
.args	ROM or RAM	.pinit	ROM or RAM
.bss	RAM	.stack	RAM
.cinit	ROM or RAM	.sysmem	RAM
.cio	RAM	.sysstack	RAM
.const	ROM or RAM	.text	ROM or RAM
.data	ROM or RAM		



Связывание


Модуль 1
012030h
011010h
01101100 PPPPPPPP PPPPPPPP PPPPPPPP
CALL P24
Абсолютное связывание P24
Модуль 2
012010h
L8: …
0102Ch = 013040h-012024h
012020h
01101101 xCCCCCCCC LLLLLLLL LLLLLLLL
BCC L16, cond
012024h
Относительное связывание L16
Модуль 3
012030h
P24: …
0DBh = 012010h-012046h
012043h
00000010E xCCCCCCCC LLLLLLLL
BCC L8, cond
012046h
Относительное связывание L8
Модуль 4
013040h
L16: …
??????h
013050h
01101100 PPPPPPPP PPPPPPPP PPPPPPPP
CALL P24X
Динамическое связывание P24X (во время загрузки)

Модели памяти

- Small (маленькая)
  память данных – до 128 кВ (16 бит адреса)
  область – до 128 кВ (16 бит адреса)

- Large (большая)
  память данных – до 16 MB (23 бита адреса)
  область – до 128 MB (16 бит адреса)

- Huge (огромная)
  память данных – до 16 MB (23 бита адреса)
  область – до 16 MB (23 бита адреса)

Подстановочные функции

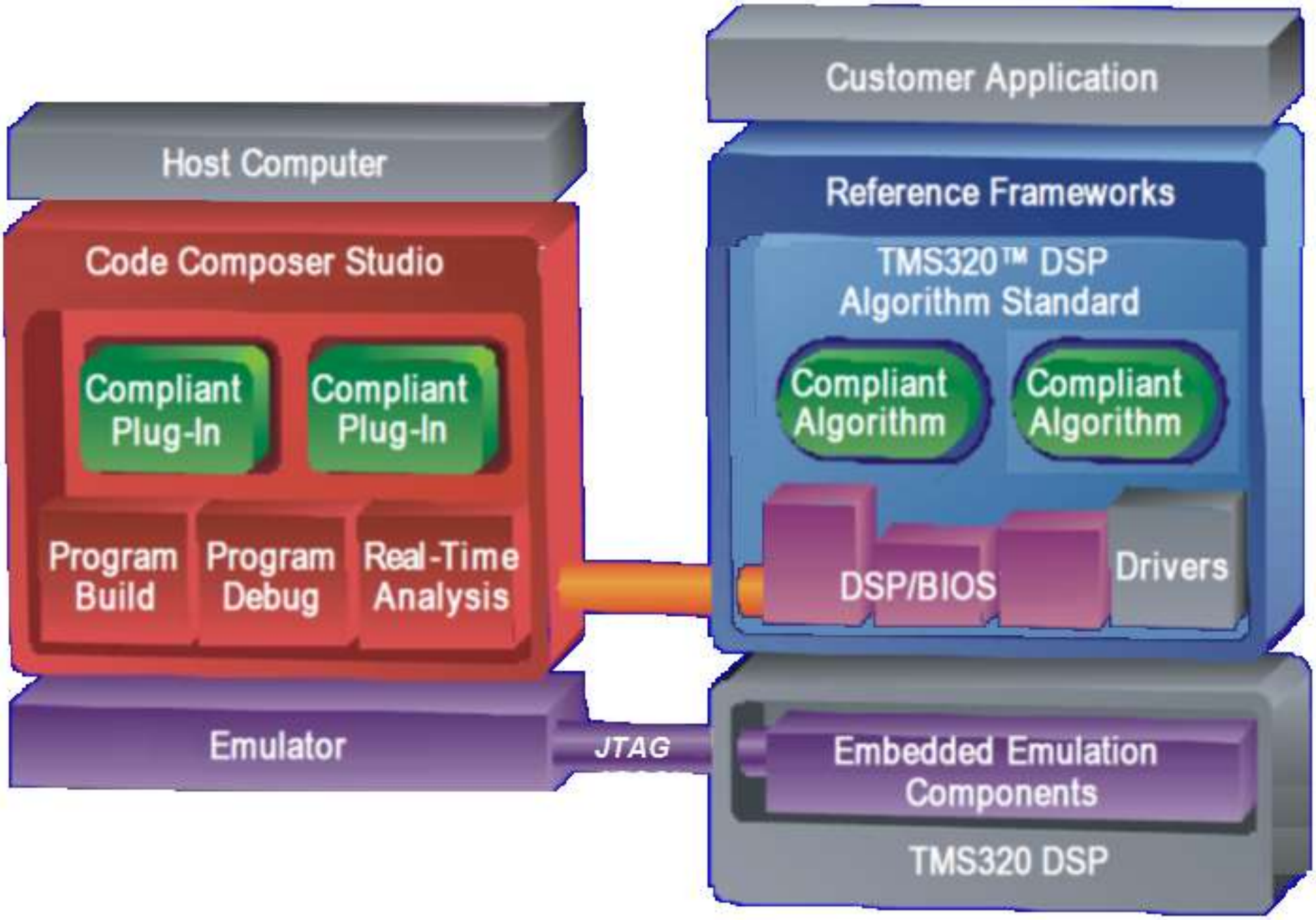
```
int         _sadd(int, int)
long        _lsadd(long, long)
long long   _llsadd(long long, long long)
int         _ssub(int, int)
long        _lssub(long, long)
long long   _llssub(long long, long long)
int         _smpy(int, int)
long        _lsmpy(int, int)
long        _smac(long, int, int)
long        _smas(long, int, int)
int         _abss(int)
long        _labss(long)
long long   _llabss(long long)
int         _sneg(int)
long        _lsneg(long)
long long   _llsneg(long long);
long        _smpyr(int, int)
long         _smacr(long, int, int)
long        _smasr(long, int, int)
```

```
int     _norm(int)
int     _lnorm(long)
long    _rnd(long)
int     _sshl(int, int)
long    _lsshl(long, int)
int     _shrs(int, int)
long    _lshrs(long, int)
```

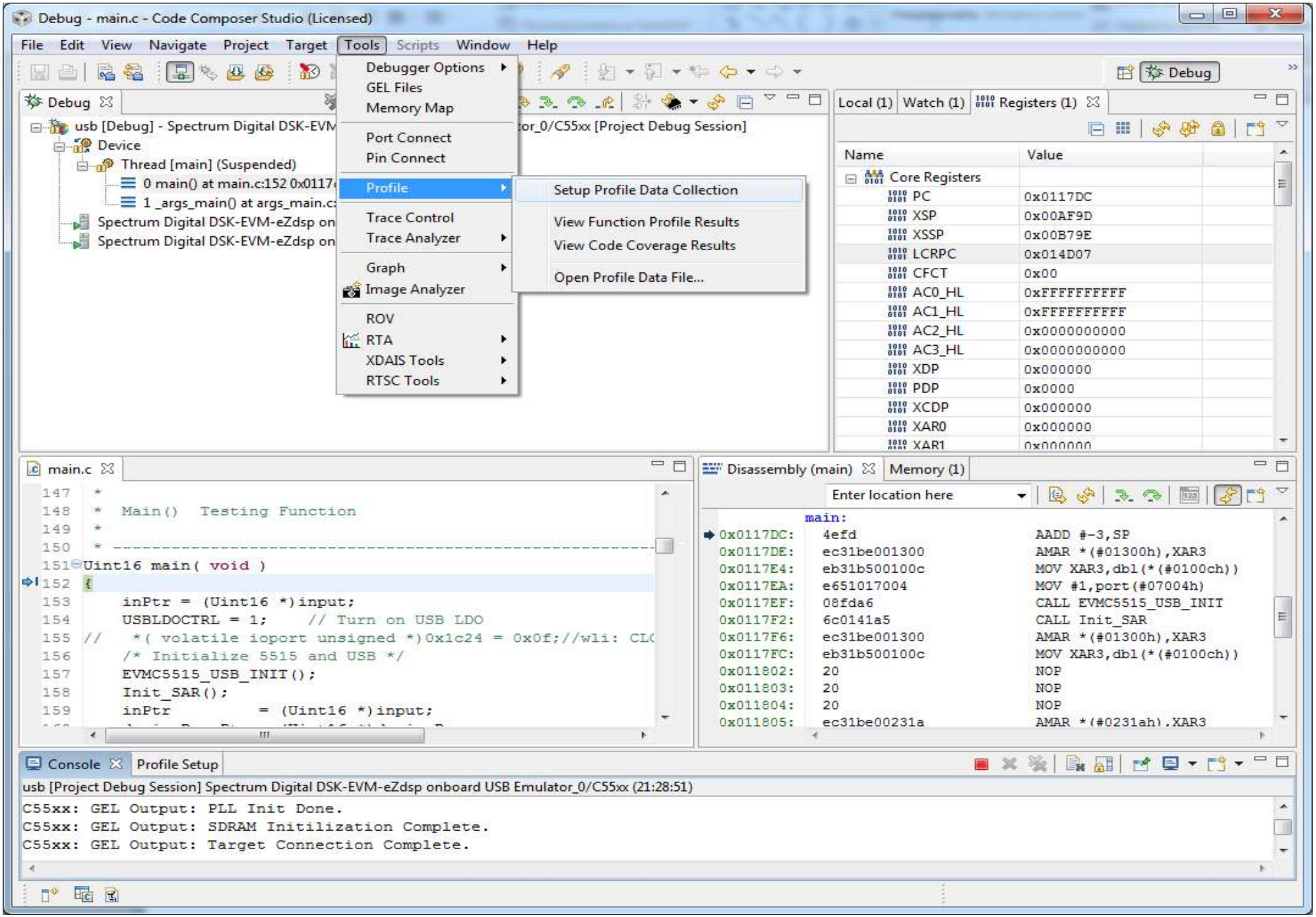
```
int sadd(int a, int b)
{
    return _sadd(a, b);
}
```

```
_sadd:
    BSET SATA
    ADD   T1, T0
    BCLR SATA
    RET
```

Технология разработки



Среда разработки



Процесс разработки