
HIGHTEC

MCS C Development Platform

BOSCH GTM Tech Day 2017

R. Knížek

“HighTec Mission is to ensure independence for the future and the most reliable and secure tools for embedded software development.

Service, customer oriented philosophy and reliable partnership are deeply engraved in our DNA.”

HighTec

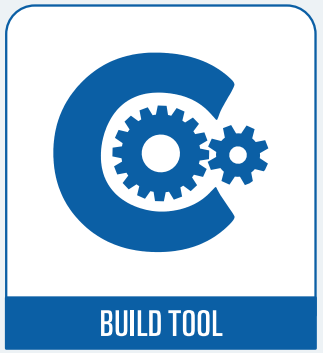
About the Company

- An independent, privately owned company, since 1982
- Specialized on Embedded Development Tools and SW solutions
- Focus on automotive and industrial markets

Any major architecture, any specialized IP is supported

HighTec Core Products

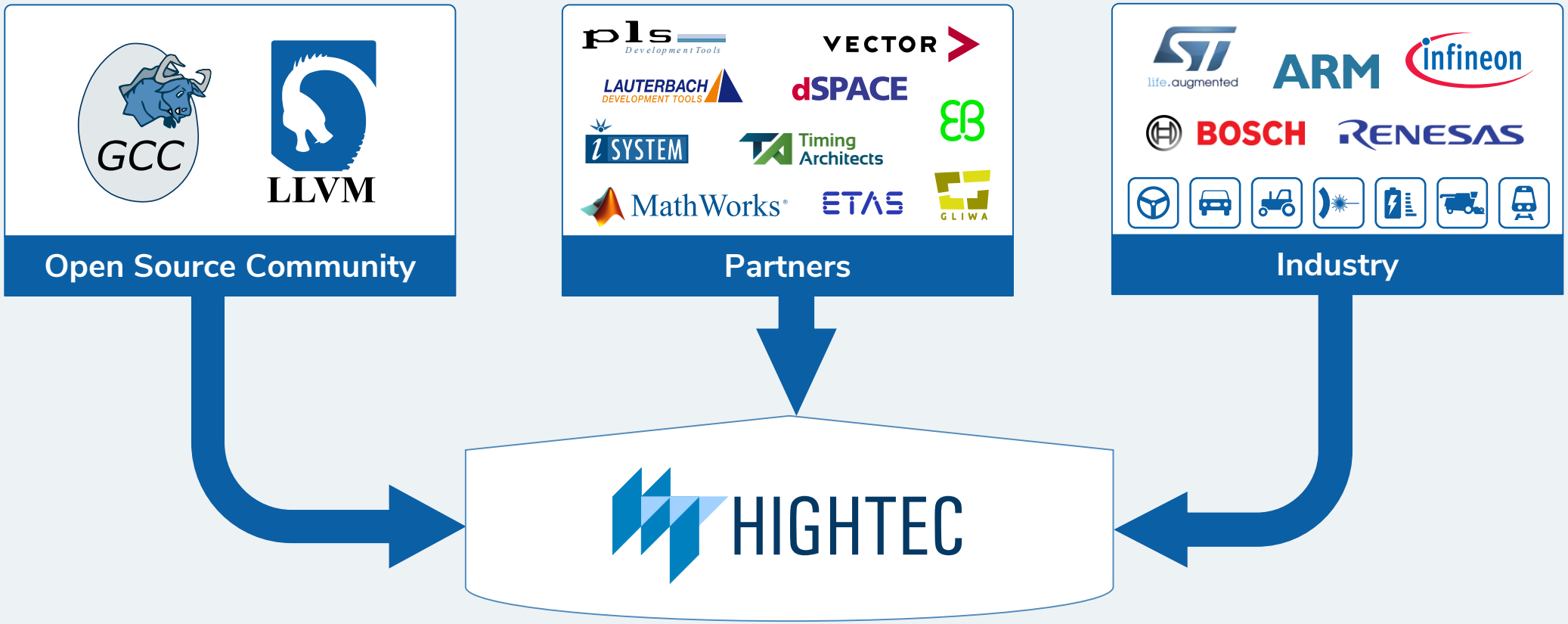
One-Stop-Shop for Embedded Tools and Development Services



- C/C++ Multiarchitecture Multicore Development Suite for ARM, AURIX, PowerPC and RH850
- PXROS-HR SIL 3 Certified Safety Multicore Real-Time Operating System
- ISO 26262 and IEC 61508 Tool Qualification for ASIL D and SIL 3 applications
- Support & Consultancy Services

Relationships

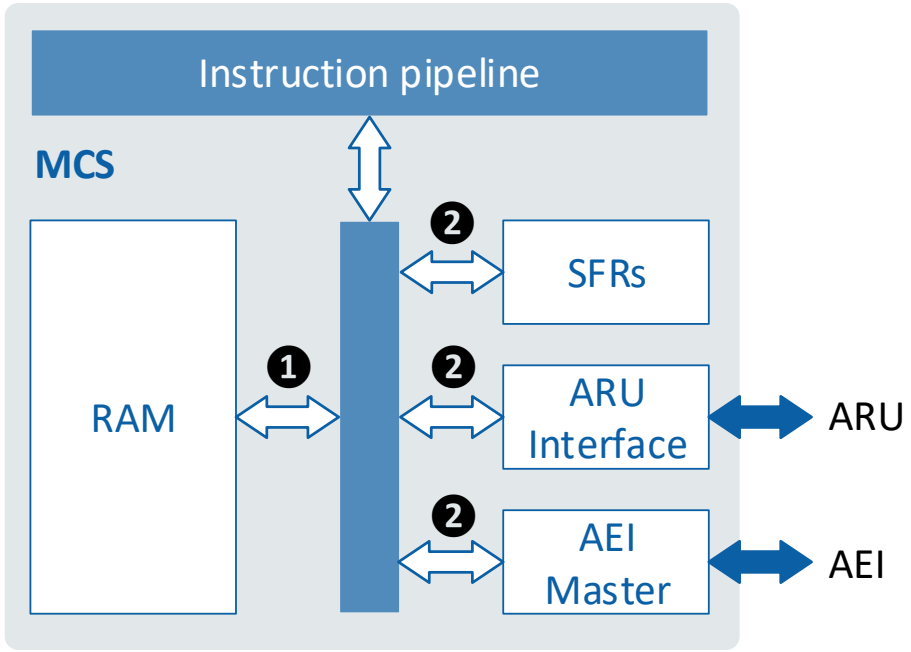
Connecting the embedded world



MCS Architectural Aspects

A challenge for both C-toolchain and application developers

- Part of a heterogeneous system
- Registers & ALU size are not 2^N
- Special data bus access & SFRs
- Special wait instructions



❶
```
int  mem_var;
```

❷
```
_aei_t aei_var;
_sfr_t STA;
```

X

```
unsigned long _brd (unsigned int addr);
unsigned int _get_STA (void);
```

MCS C Compiler Overview

Language aspects

ANSI/ISO C99

Native Data Types

24 and 48 bits

Built-in Intrinsic Functions

SFR, WAIT, ARU/AEI

Extensive Link Command Language

Latest GTM-IP v3.x

Including v1.x and 2.x assembler

3rd party Debugger Support

Lauterbach, pls, iSystem

Advanced Optimizations

Size vers. speed performance balance
Target-specific options

MCS C Compiler

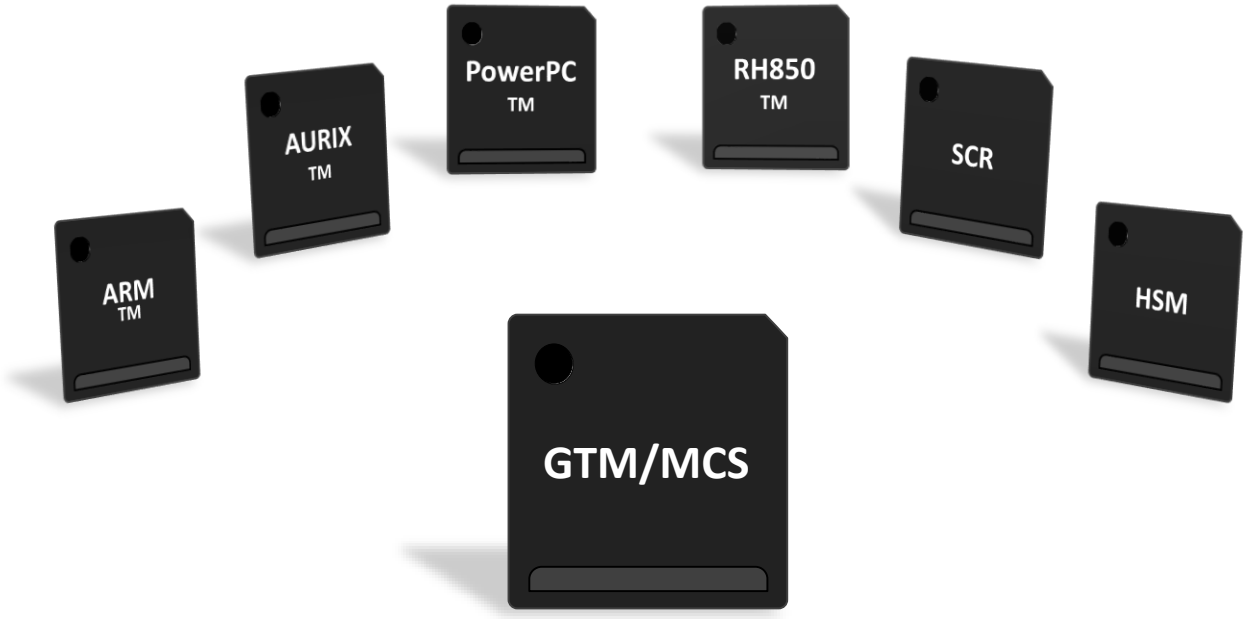
The environment

- Facilitated Casper to HighTec assembler conversion
- Validated against industry-standard C99 test suites
- Long-term availability
- Frozen Version support

- Familiar LLVM/GCC interface
- Industry's fastest build system

MCS C Development Suite

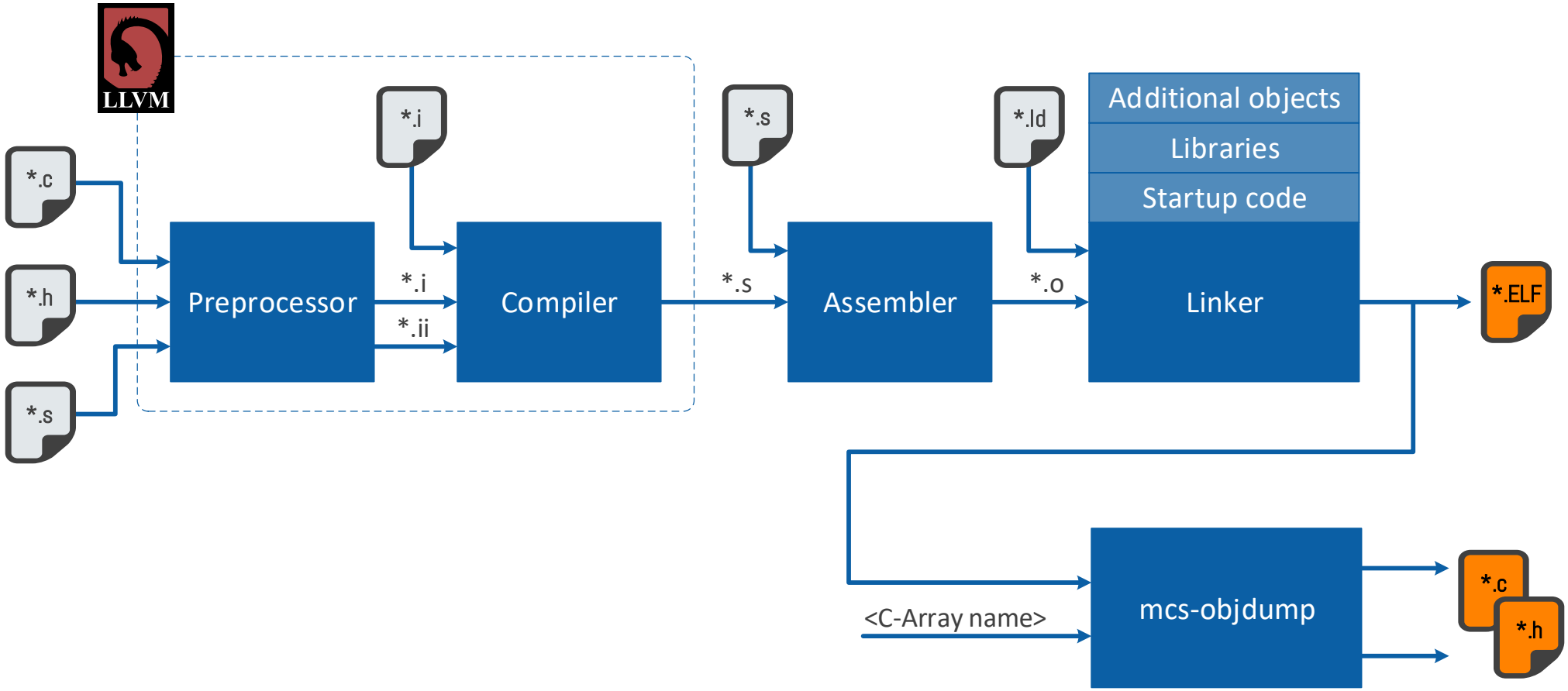
A part of the HighTec's toolchain family



Heterogeneous multicore linker support – Cross-Linking

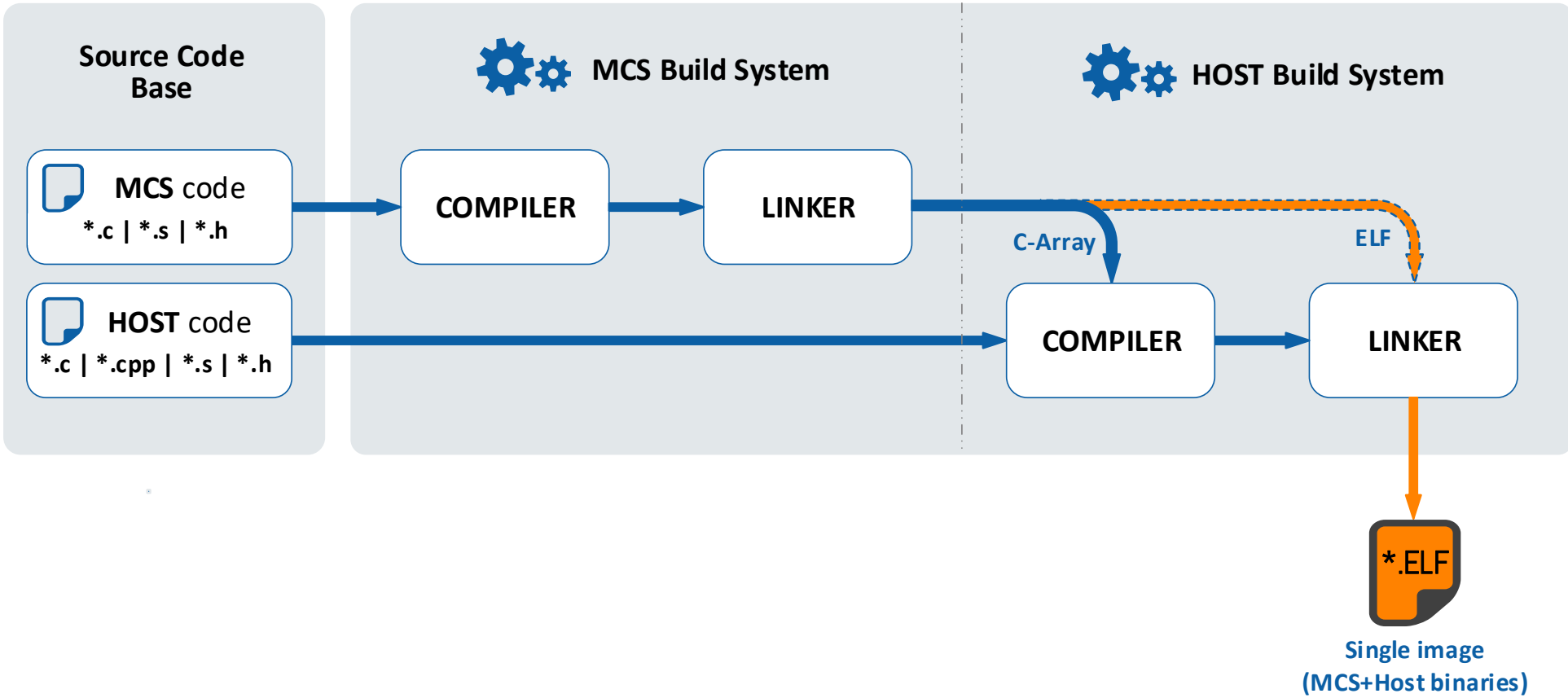
MCS Source Code Build Process

MCS Development Suite



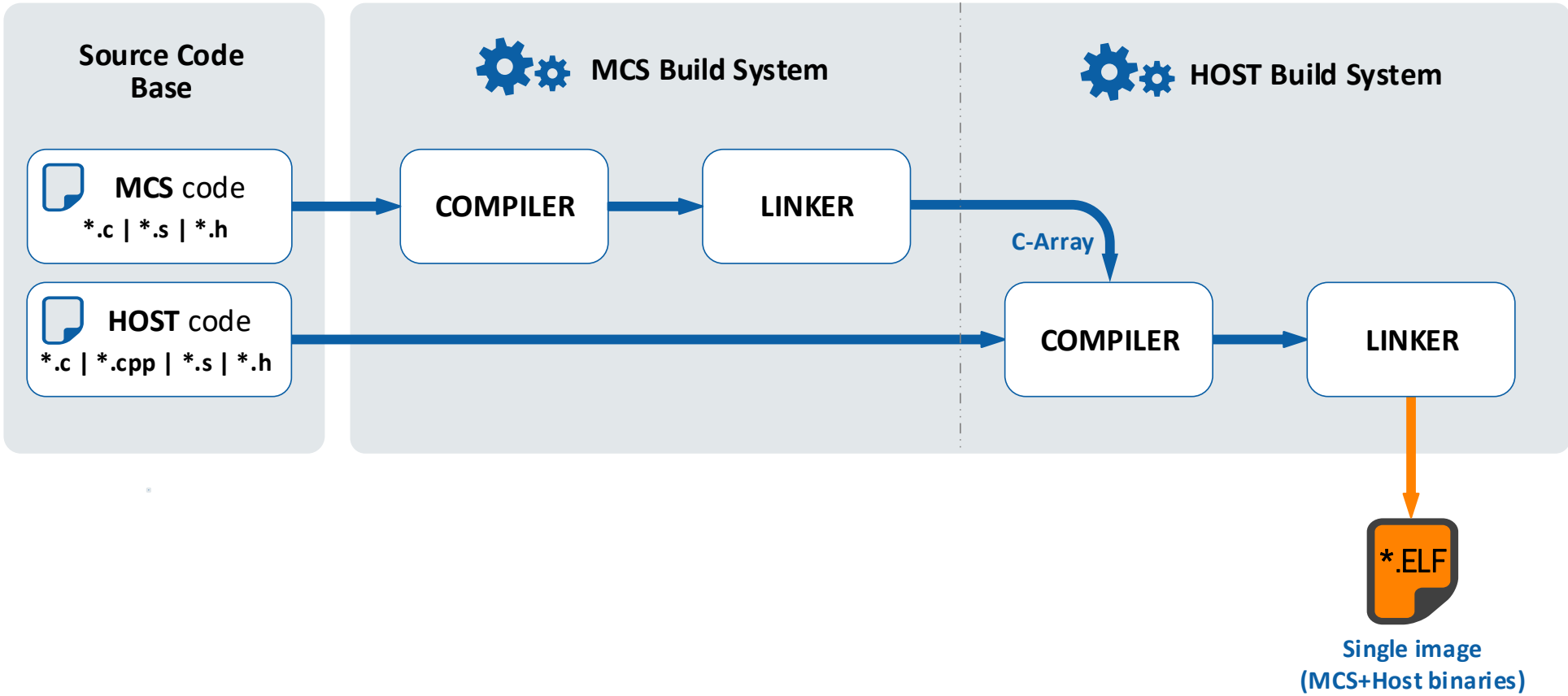
MCS & Host Build

A two-step process



MCS & Host Build: C-Array

MCS C-Array source files compiled together with the Host source

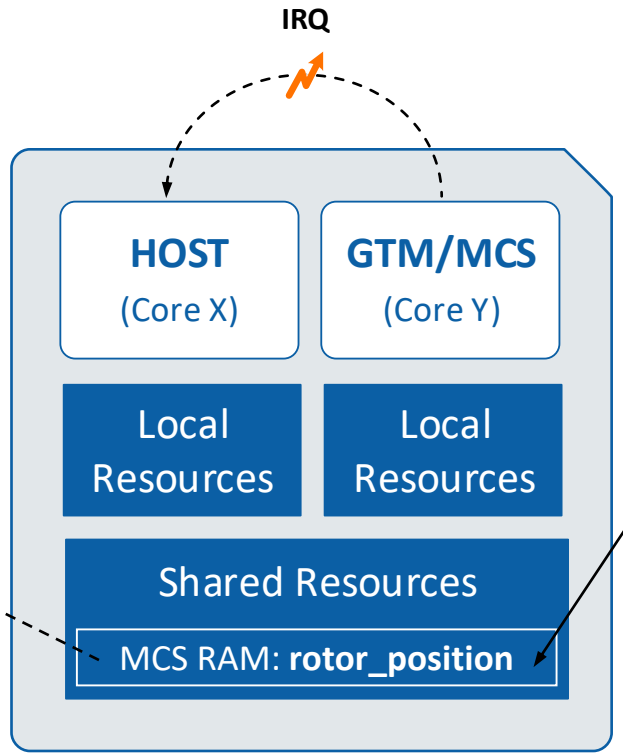


Example

Exporting MCS variables to the Host code

HOST Source Code (compiler **X**)

```
void ISR_MCS_Channel_0(void)
{
    …

    Do_Something (*rotor_position);

    …
}
```

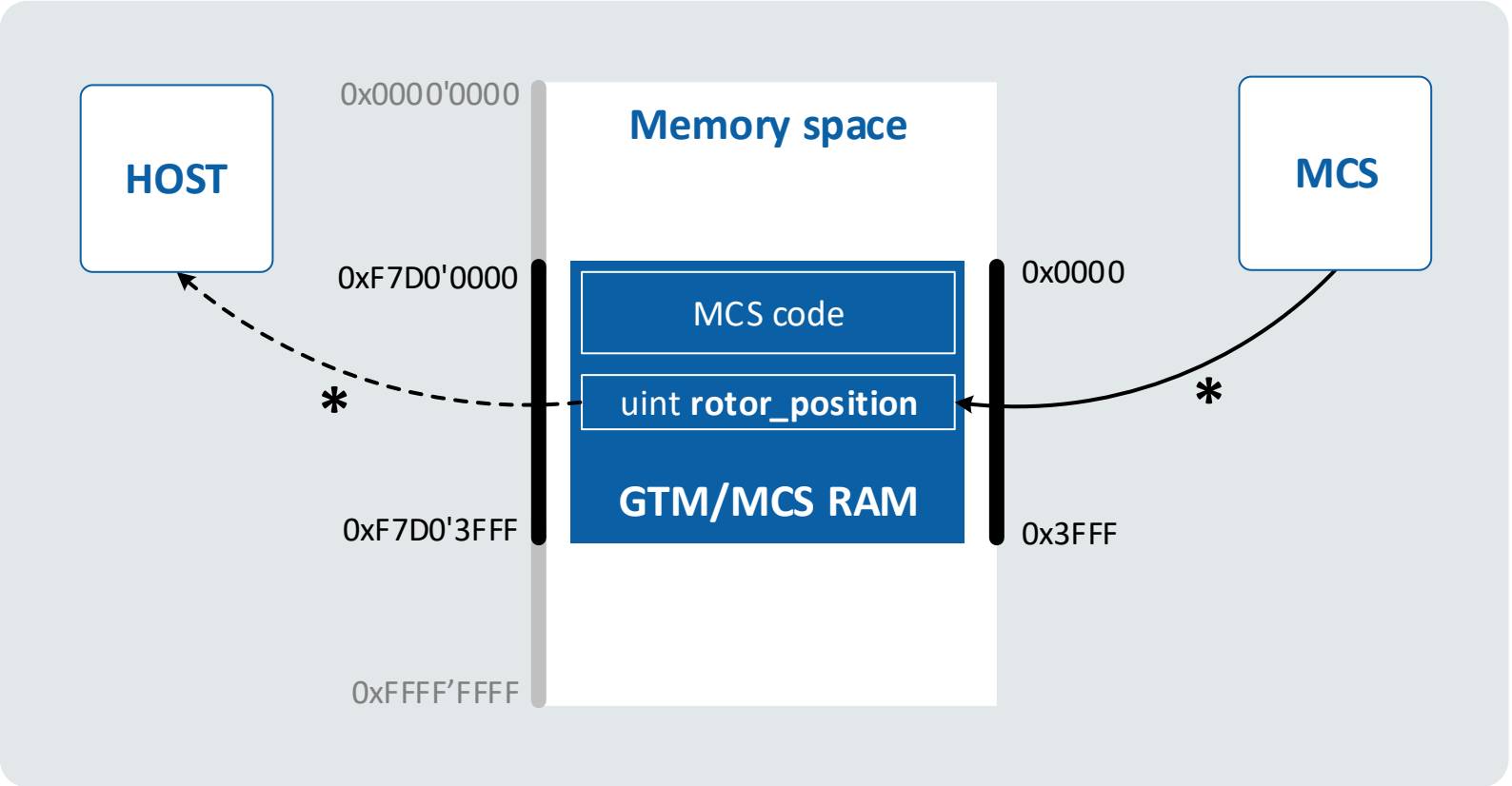


GTM/MCS Source Code (compiler **Y**)

```
unsigned int rotor_position;

int Main_Channel_0(void)
{
    …

    rotor_position = Calc_Value();

    STA.IRQ = 1;
    …
}
```

GTM/MCS RAM Access

MCS versus Host access to the MCS RAM



&rotor_position = 0xF7D0**'03FC**

&rotor_position = 0x**03FC**

MCS Code Image as a C-Array

The files become part of the Host application source code

mcs00.c *.c

```
#include "mcs00.h"

unsigned long mcs00[263] = {
/* Section: .mcs_start */
/* _start [0]: */
    0xE0000020,
    0xE0000054,
    0xE0000088,
    0xE00000BC,
    0xE00000F0,
    0xE0000124,
    0xE0000158,
    0xE000018C,
/* mcs_start_ch0 [8]: */
    0xA001041C,
    0x20000001,
    0xA002041C,

    …
};
```

mcs00.h *.h

```
#ifndef mcs00_h_
#define mcs00_h_

extern unsigned long mcs00[263];

#define OFFSET_mcs00                    (0)
#define SIZE_mcs00                      (1052)

#define LABEL_mcs00_mcs_start_ch0       (8)
#define LABEL_mcs00_mcs_exit_ch0        (20)
#define LABEL_mcs00_mcs_main_ch0        (131)
…

#define LABEL_mcs00_rotor_position      (255)
#define LABEL_mcs00_SYSTEM_STACK_CH0    (271)

…
#endif
```

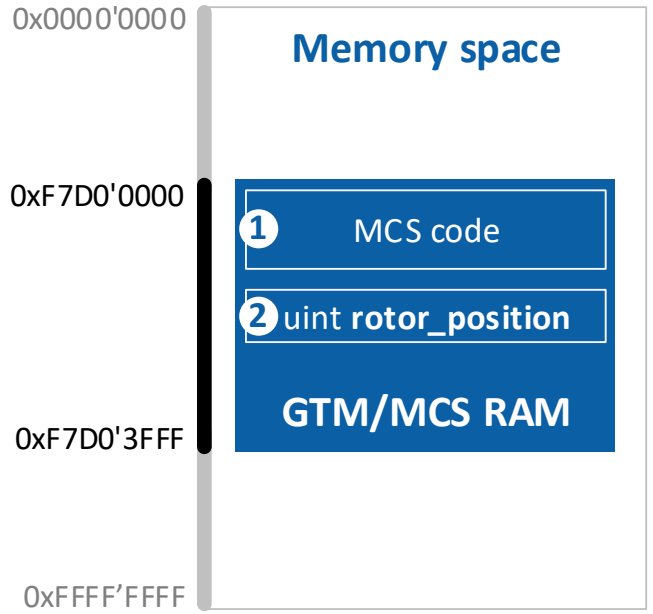
C-Array - Integration

Handling of the MCS image in the Host application code

1 Copy the MCS image to the MCS RAM

Include the `mcs00.h`

Copy the `unsigned long mcs00[…]`



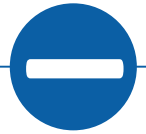
2 Declare links to the MCS export variables

```
uint *rotor_position = (uint*) (0xF7D00000 +
                       (LABEL_mcs00_rotor_position * 4));
```

The C-Array Summary

Benefits and limitations

- Standard C source
- Compiler vendor independent

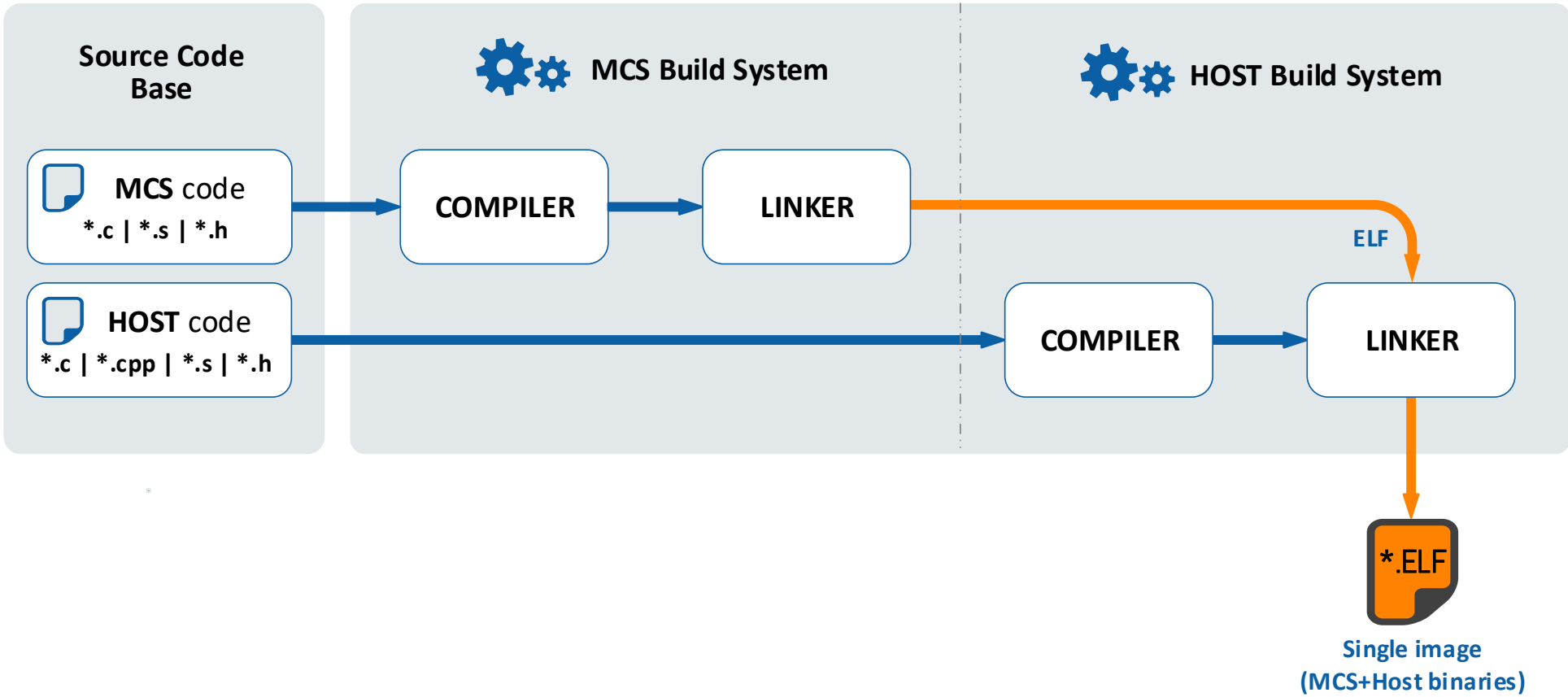


- Host integration code required
- Symbol/debug info is lost
- Explicit MCS variable pointers
- No compiler/linker warnings
- Increased build time

Difficult to keep consistency between Host and MCU sources

MCS & Host Build: Cross-Linking

MCS object image linked together with the Host object files



Cross-Linking

Unified heterogeneous multicore development

- The MCS output ELF object file is merged directly with the Host application, during its linking stage – a single final ELF is created
- Resolves symbols and cross-references from heterogeneous object files produced by compilers for different architectures

Enabled through the unique synergy among HighTec's toolchain product family

Cross-Linking – Integration

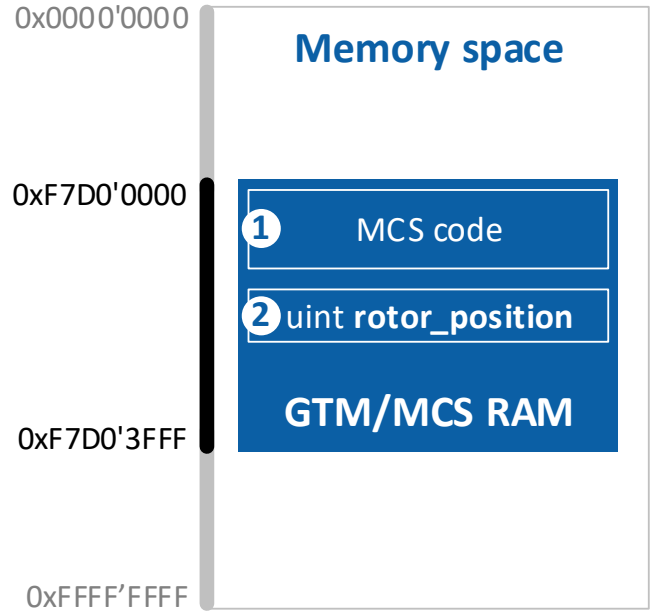
Handling of the MCS image in the Host application code

1 Copy the MCS image to the MCS RAM

None. (Implicit C-init clear/copy table process.)

2 Declare links to the MCS export variables

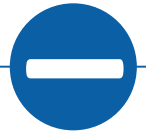
```
extern uint rotor_position;
```



Cross-Linking Summary

Benefits and limitations

- MCS symbol info is preserved
- MAP file with cross-references
- Implicit refs. to MCS variables
- Link-time consistency checks
- Instrumented MCS code load

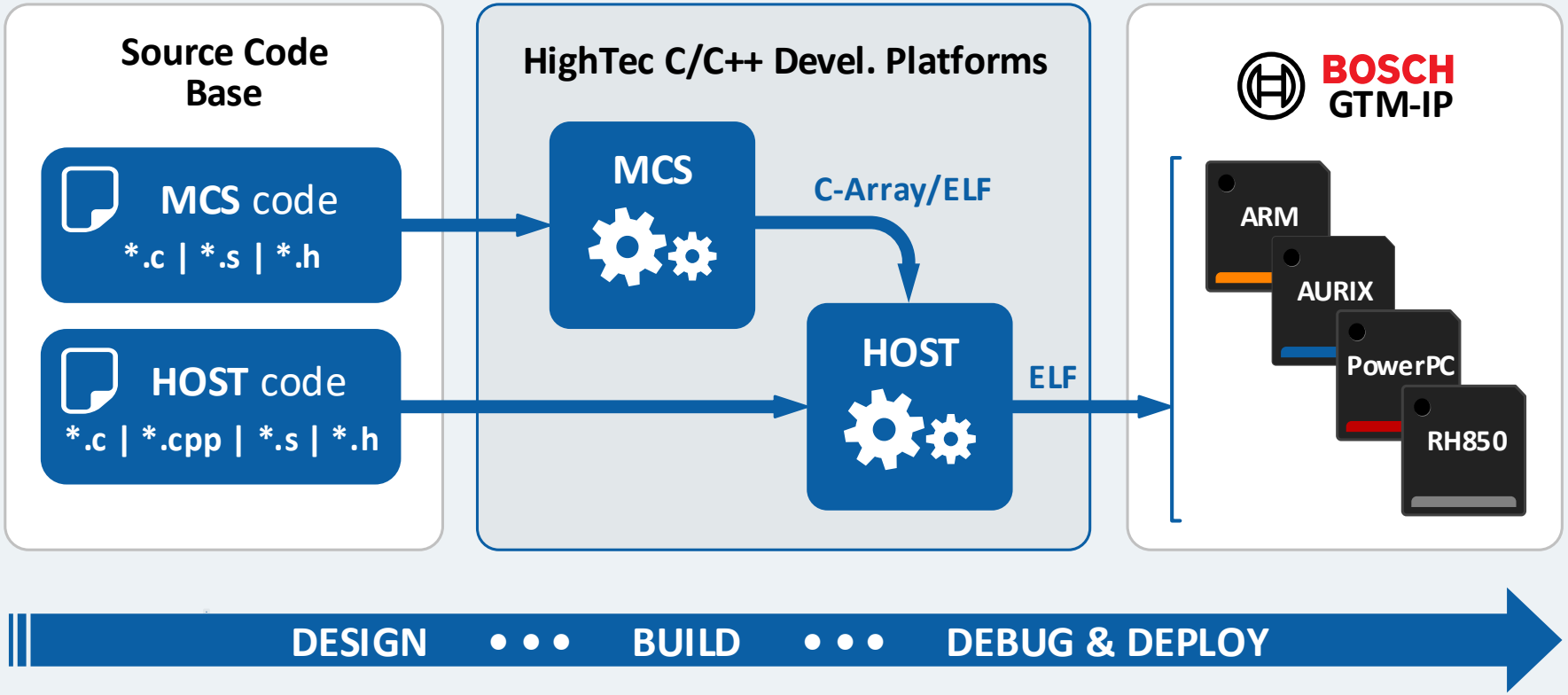


- May increase linker file complexity

Supports the true unified heterogeneous multicore development

HighTec's Unified Development

GTM/MCS in the heterogeneous multicore environment



Need More Information?

Let us hear You!

www.hightec-rt.com

support@hightec-rt.com

sales@hightec-rt.com


HIGHTEC