Module 6

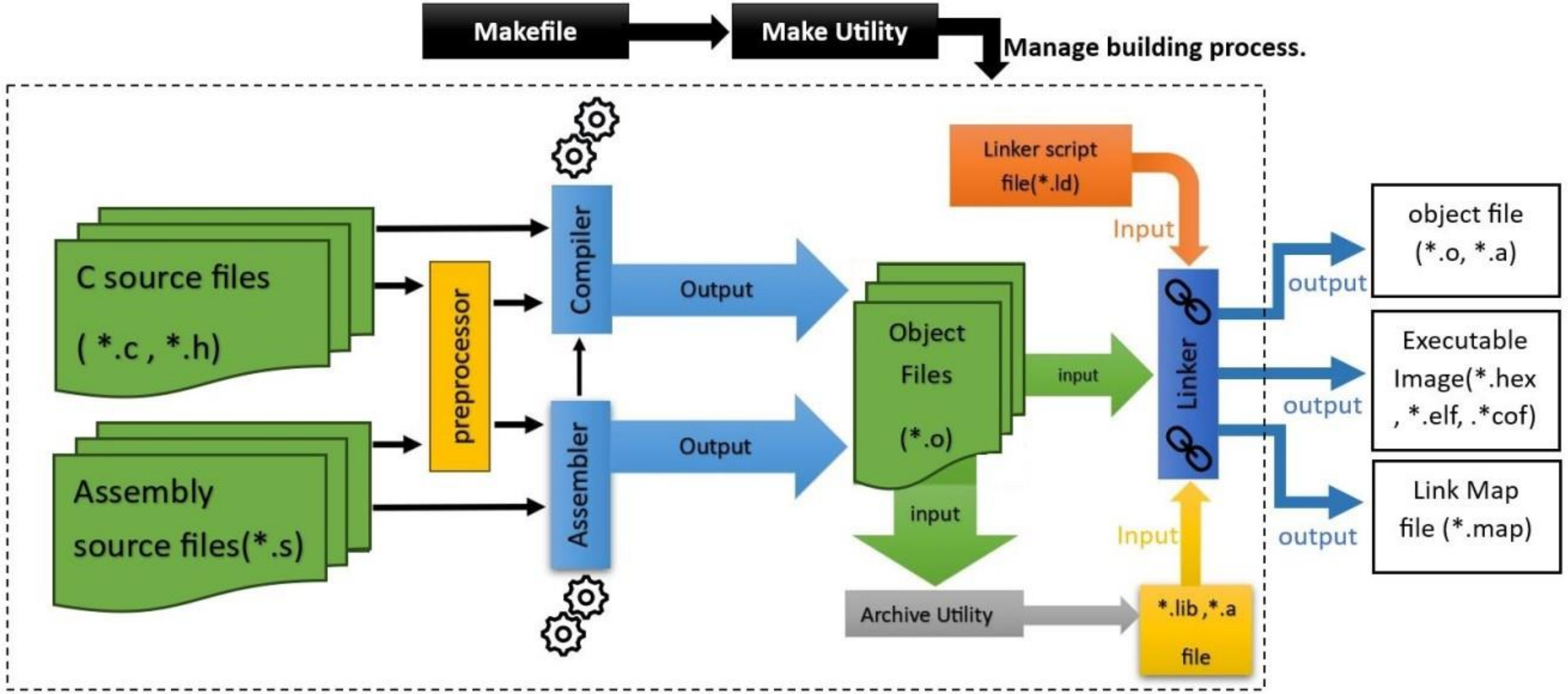
ToolChain

GNU Arm Embedded Toolchain

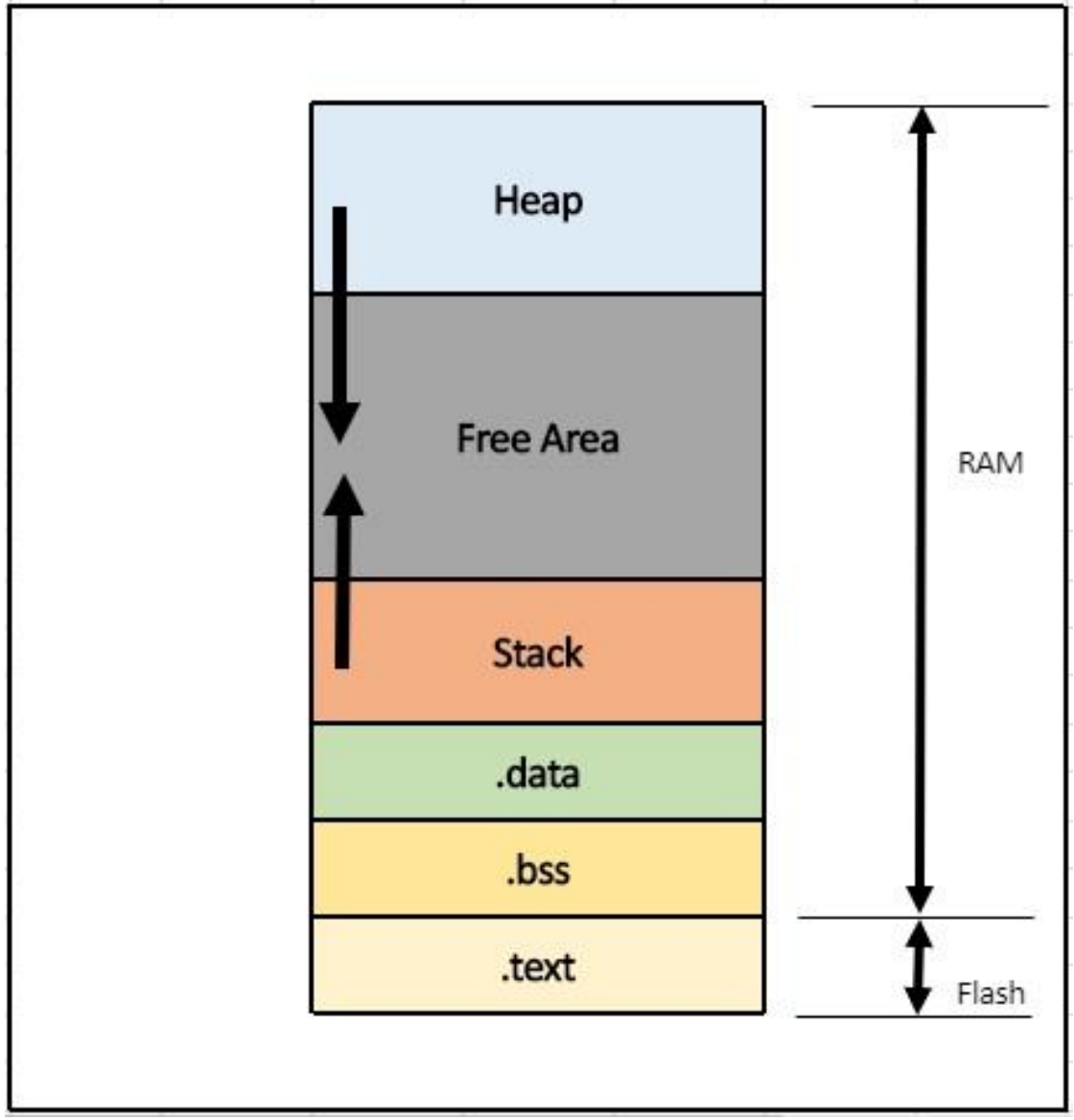
Thynk Loop

	Tool	Used for
Build tools	**arm-none-eabi-gcc**	Compiling code into object files, and it has linker internally to link object files into an executable.
	arm-none-eabi-ld	Links object files into an executable. (called internally by arm-none-eabi-gcc)
	arm-none-eabi-ar	Archives multiple object files into static library.
Binary Utilities	**arm-none-eabi-readelf**	Read the content of ELF file.
	arm-none-eabi-objdump	Read the internals of ELF file.
	arm-none-eabi-nm	Read the symbols inside an object file or executable.
	arm-none-eabi-strings	Read text strings inside a binary file.
	arm-none-eabi-strip	Strip the binary file from some optional sections.
	arm-none-eabi-addr2line	Converts an address in the binary to a source file name and line number.
	arm-none-eabi-size	Display the ELF file section sizes and total size.
Debugger	**arm-none-eabi-gdb**	Debugger

GNU Arm Embedded Toolchain



Memory Layout



Linker File

Thynk Loop

- Entry Point
- Stack Configuration
- Memory Configuration
- Sections

Custom - Memory Configuration

```
MEMORY
{
        RAM    (xrw)    : ORIGIN = 0x20000000, LENGTH = 64K
        FLASH  (rx)     : ORIGIN = 0x8000000, LENGTH = 256K
        USER (xrw)    : ORIGIN = 0x2000F000, LENGTH = x K
}
```

Custom - Section Configuration

```
SECTIONS
{
  .custom_data :
  {
    . = ALIGN(4);
    *(.custom_data)  /* Place all `.custom_data` sections here */
    . = ALIGN(4);
  } >USER
}
```

Custom – Variable Implementation 1

Thynk Loop

```
__attribute__((section(".custom_data")))
uint32_t custom_variable = 0; // Variable placed in .custom_data
```

Custom – Variable Implementation 3

```
#define CUSTOM_SECTION __attribute__((section(".custom_data")))

CUSTOM_SECTION uint32_t custom_variable1 = 0x12345678;
```

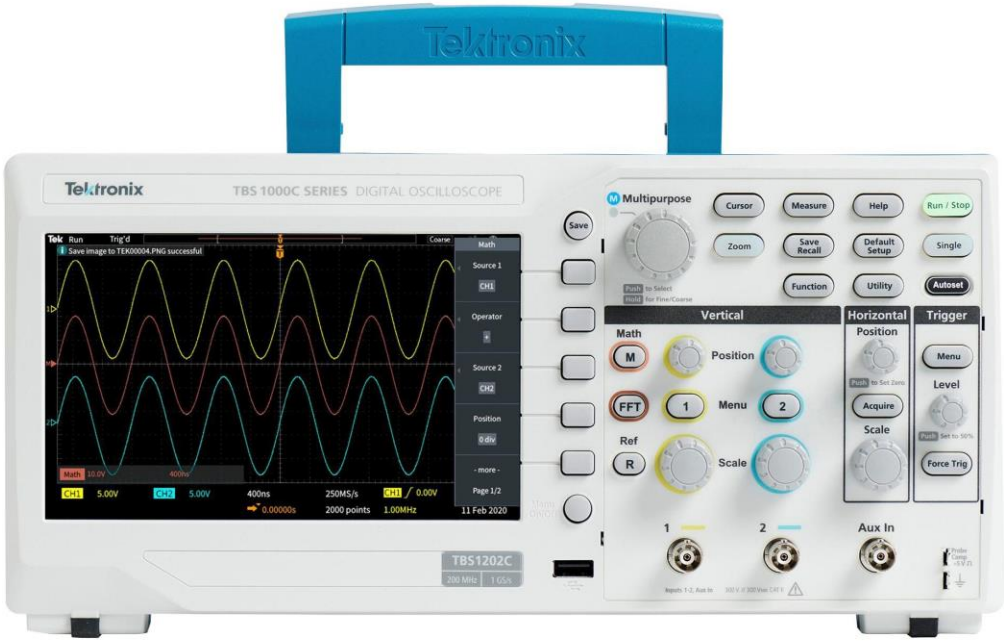
Custom – Variable Implementation 3

```
#pragma section(".custom_data")

__attribute__((section(".custom_data")))
uint32_t custom_variable1 = 0x12345678;
uint32_t custom_variable2 = 0x87654321;
uint16_t custom_variable3 = 0xABCD;
uint8_t custom_variable4 = 0x12;

#pragma section() // End section
```

Debugging Tools

PRINTF

Debugger



GPIO

GPIO

Register	Purpose	Width
GPIOx_MODER	Configures pin mode (input/output/alternate/analog).	32 bits
GPIOx_OTYPER	Configures output type (push-pull/open-drain).	16 bits
GPIOx_OSPEEDR	Configures output speed (low/medium/high/very high).	32 bits
GPIOx_PUPDR	Configures pull-up/pull-down resistors.	32 bits
GPIOx_IDR	Reads input state of pins.	16 bits
GPIOx_ODR	Writes output state of pins.	16 bits
GPIOx_BSRR	Atomic bit set/reset for pins.	32 bits
GPIOx_LCKR	Locks pin configuration.	16+1 bits
GPIOx_AFRL/AFRH	Configures alternate functions.	32 bits

GPIO

Type	High State	Low State	Use Cases
Push-Pull	Actively driven by GPIO pin	Actively driven by GPIO pin	Driving LEDs, SPI communication.
Open-Drain	Via external pull-up resistor	Actively driven by GPIO pin	I^2C, interfacing with higher voltage levels.
Pull-Up	Via resistor to Vcc	No pull (floating or open)	Default HIGH for input or open-drain pins.
Pull-Down	No pull (floating or open)	Via resistor to ground	Default LOW for input or open-drain pins.
Floating	Undefined (noise-prone)	Undefined (noise-prone)	Rarely used unless necessary.
Analog	Analog signal (varies)	Analog signal (varies)	ADC/DAC and sensor interfaces.

GPIO

Thynk Loop

Bits	Speed Setting	Approximate Speed
00	Low speed	Up to 2 MHz
01	Medium speed	Up to 25 MHz
10	High speed	Up to 50 MHz
11	Very high speed	Up to 100 MHz