STM32 as GPU

Today a lot of code is written for AVR (Arduino), and we all know limitations of these MCUs, for example extremely slow drawing on LCD screens, not enough ROM, huge SD libs and other!

But what if You want to use these MCUs longer, and You absolutely doesn’t want to waste time for porting sources, looking for hundreds of mans and investigate STM32?

That is why this GPU coprocessor project was created!

Created for simple MCU\_CPU (like AVR) and help to accelerate 2d graphics on LCDs by using powerful MCU like ARM STM32.

For maximum simplicity USART was selected as main communication interface. There are tons of realizations of this interface (as hardware as software) for all type of MCUs.

Flexible Baud rate selection allows matching from low speed to hi speed transfer (some MCUs has no resonators and required low speed).

Only two lines is used to communicate (even one line possible).

GPU coprocessor can save ROM space, RAM, free MCU\_CPU from executing code for LCD and left more resources for main application!

Just send commands to USART like it`s an LCD and GPU do it`s job.

MCU\_CPU lib is written on C, so it can be simply ported to where You want!

All this allow to use MCU\_CPU even like AVR ATtiny family, isn’t cool huh?

General Description

Tiles:

* up to 96 tiles 8x8;
* up to 16 tiles 8x16;
* up to 8 tiles 16x16;
* tiles group (metatile) up to 8 tiles;
* any colors per sprite from palette of 80 colors;
* add tiles;
* delete tiles;
* draw tiles;

Draw functions:

* almost all from GFX lib;
* GUI;

Interface:

* USART to interface GPU (1200, 2400, 9600, 57600, 115200, 1M) ;
* SPI2 to interface SD;
* Paralel (8080) to interface LCD (STM32F103VET6);
* SPI1 to interface LCD (STM32F103C8T6);

Input GPU buffer:

* at least 200 commands (depend on size of commands and type of MCU RAM);

Data packet size:

* CMD consume 1 byte;
* DATA consume max 6\*uint16\_t bytes;

MCU:

* STM32F103C8T6 64 Flash, 20 RAM;
* STM32F103VET6 512 Flash, 64 RAM;

DMA supported by:

* STM32F103C8T6 64 Flash, 20 RAM;
* STM32F103VET6 512 Flash, 64 RAM;

FSMC:

* only by STM32F103VET6 512 Flash, 64 RAM;

GPIO Pinout

CPU\_GPU interface:

|  |  |  |  |
| --- | --- | --- | --- |
| Name In Code | MCU periph name | GPIO port | GPIO number |
| - | USART1\_RX | GPIOA | GPIO\_Pin\_10 |
| - | USART1\_TX | GPIOA | GPIO\_Pin\_9 |
| BAUD\_9600\_PIN | -not yet ready- |  |  |
| BAUD\_57600\_PIN | -not yet ready- |  |  |
| BAUD\_115200\_PIN | -not yet ready- |  |  |
| BAUD\_1M\_PIN | -not yet ready- |  |  |

SD Card:

|  |  |  |  |
| --- | --- | --- | --- |
| Name In Code | MCU periph name | GPIO port | GPIO number |
| GPIO\_Pin\_SPI\_SD\_MISO | SPI2\_MISO | GPIOB | GPIO\_Pin\_14 |
| GPIO\_Pin\_SPI\_SD\_MOSI | SPI2\_MOSI | GPIOB | GPIO\_Pin\_15 |
| GPIO\_Pin\_CS | SPI2\_CS | GPIOB | GPIO\_Pin\_12 |
| GPIO\_Pin\_SPI\_SD\_SCK | SPI2\_SCK | GPIOB | GPIO\_Pin\_13 |

LCD (SPI1):

|  |  |  |  |
| --- | --- | --- | --- |
| Name In Code | MCU periph name | GPIO port | GPIO number |
| MOSI\_PIN | SPI1\_MOSI | GPIOA | GPIO\_Pin\_7 |
| TFT\_SS\_PIN | SPI1\_CS | GPIOB | GPIO\_Pin\_10 |
| SCK\_PIN | SPI1\_SCK | GPIOA | GPIO\_Pin\_5 |
| TFT\_DC\_PIN | - | GPIOB | GPIO\_Pin\_11 |
| TFT\_RES\_PIN | - | GPIOB | GPIO\_Pin\_1 |

LCD (8080):

|  |  |  |  |
| --- | --- | --- | --- |
| Name In Code | MCU periph name | GPIO port | GPIO number |
| FSMC\_PIN\_D0 | D0 | GPIOD | GPIO\_Pin\_14 |
| FSMC\_PIN\_D1 | D1 | GPIOD | GPIO\_Pin\_15 |
| FSMC\_PIN\_D2 | D2 | GPIOD | GPIO\_Pin\_0 |
| FSMC\_PIN\_D3 | D3 | GPIOD | GPIO\_Pin\_1 |
| FSMC\_PIN\_D4 | D4 | GPIOE | GPIO\_Pin\_7 |
| FSMC\_PIN\_D5 | D5 | GPIOE | GPIO\_Pin\_8 |
| FSMC\_PIN\_D6 | D6 | GPIOE | GPIO\_Pin\_9 |
| FSMC\_PIN\_D7 | D7 | GPIOE | GPIO\_Pin\_10 |
| FSMC\_PIN\_D8 | D8 | GPIOE | GPIO\_Pin\_11 |
| FSMC\_PIN\_D9 | D9 | GPIOE | GPIO\_Pin\_12 |
| FSMC\_PIN\_D10 | D10 | GPIOE | GPIO\_Pin\_13 |
| FSMC\_PIN\_D11 | D11 | GPIOE | GPIO\_Pin\_14 |
| FSMC\_PIN\_D12 | D12 | GPIOE | GPIO\_Pin\_15 |
| FSMC\_PIN\_D13 | D13 | GPIOD | GPIO\_Pin\_8 |
| FSMC\_PIN\_D14 | D14 | GPIOD | GPIO\_Pin\_9 |
| FSMC\_PIN\_D15 | D15 | GPIOD | GPIO\_Pin\_10 |
| FSMC\_PIN\_WR | NWE | GPIOD | GPIO\_Pin\_5 |
| FSMC\_PIN\_RD | NOE | GPIOD | GPIO\_Pin\_4 |
| FSMC\_PIN\_CS | NE1 | GPIOD | GPIO\_Pin\_7 |
| FSMC\_PIN\_RS | A16 | GPIOD | GPIO\_Pin\_11 |
| TFT\_RES\_PIN |  |  |  |

Some Math

HEAP size is limited and share to all type of tiles!

uint16\_t lastTile16x16[256]; // 512

uint16\_t lastTile8x16[128]; // 256

uint16\_t lastTile8x8[64]; // 128

// total: 896

// If RAM is 10240 then:

8 tiles if 16x16;

16 tiles if 8x16;

96 tiles if 8x8;

each 16x16 consume as 4 8x8

each 8x16 consume as 2 8x8

160x128 px

20x16 | 8x8

10x8 | 16x16

20x16 = 320 bytes for one screen

320x240 px

40x30 | 8x8

20x15 | 16x16

40x30 = 1200 bytes for one screen

Memory map:

10240 bytes for tiles;

3072 bytes for USART command buffer;

1024 bytes for stack;

363 bytes WTF;

Free RAM:

5781 bytes

// if tiles 2 bytes

4 x (16x16) = 2048

8 x (8x16) = 2048

48x (8x8) = 6144

// if tiles 1 bytes

8 x (16x16) = 2048

16 x (8x16) = 2048

96 x (8x8) = 6144