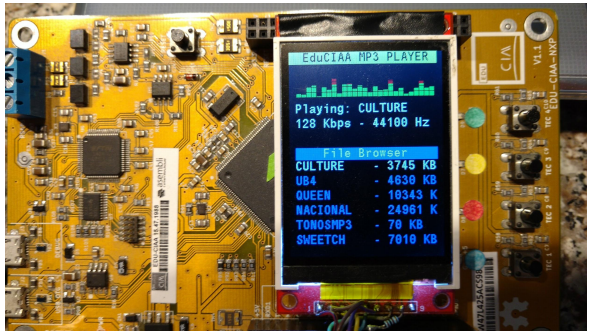
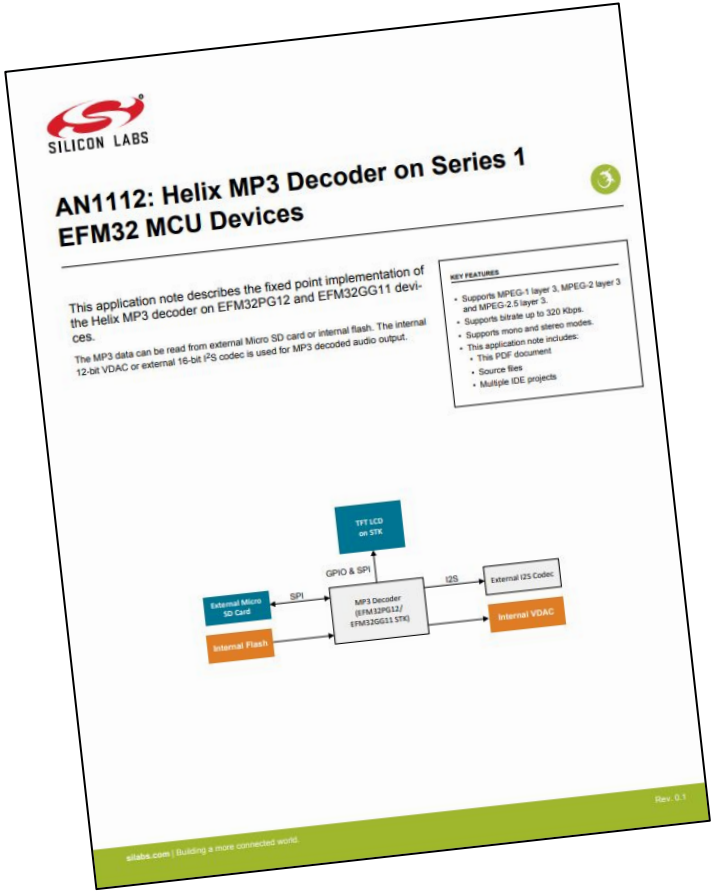
TP Final
Protocolos
CESE 6Co2018

Reproductor de MP3 con EduCIAA

Telmo Moya Grondona



Helix MP3 Decoder



SILICON LABS

AN1112: Helix MP3 Decoder on Series 1 EFM32 MCU Devices

This application note describes the fixed point implementation of the Helix MP3 decoder on EFM32PG12 and EFM32GG11 devices.

The MP3 data can be read from external Micro SD card or internal flash. The internal 12-bit VDAC or external 16-bit I²S codec is used for MP3 decoded audio output.

KEY FEATURES

- Supports MPEG-1 layer 3, MPEG-2 layer 3 and MPEG-2.5 layer 3.
- Supports bitrate up to 320 Kbps.
- Supports mono and stereo modes.
- This application note includes:
  - This PDF document
  - Source files
  - Multiple IDE projects

Rev. 0.1

silabs.com | Building a more connected world.

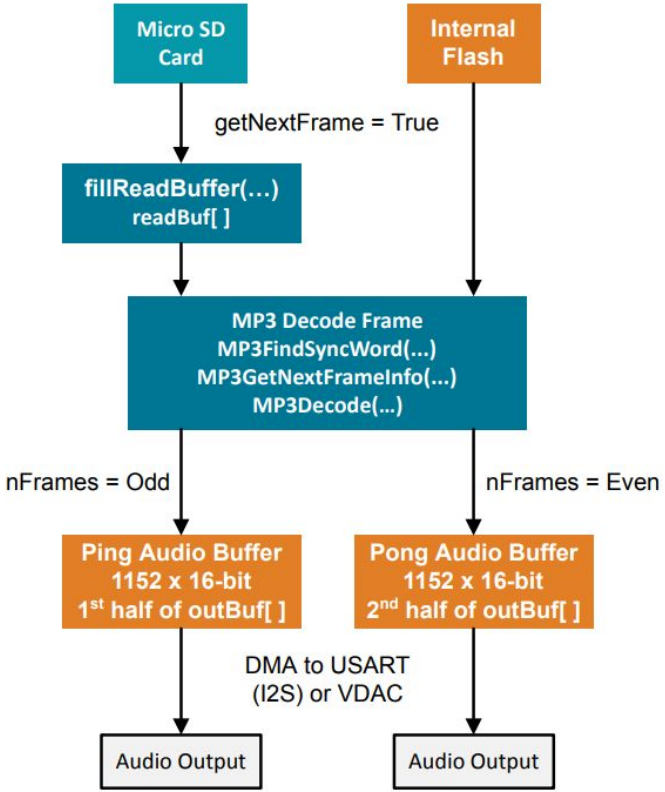
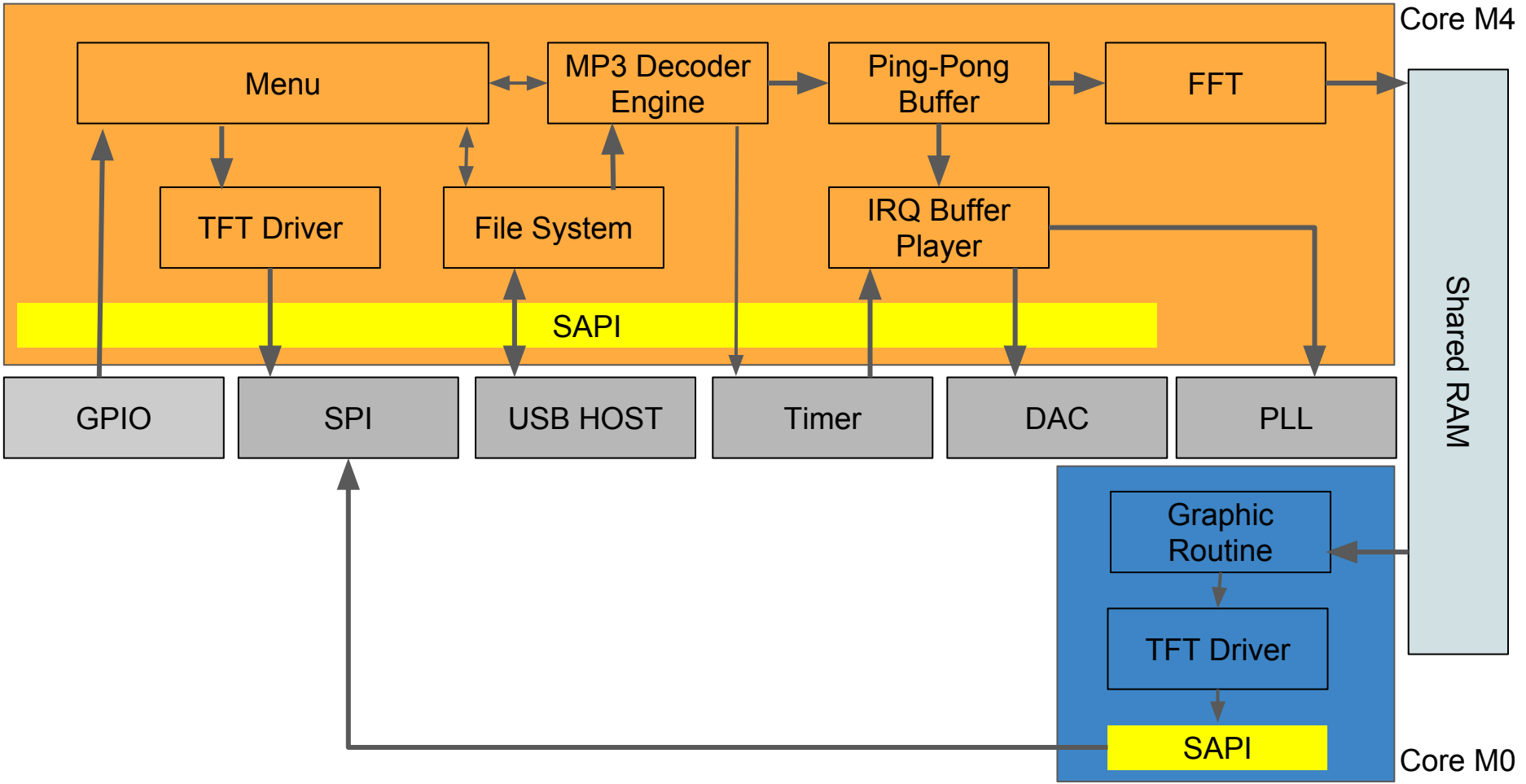
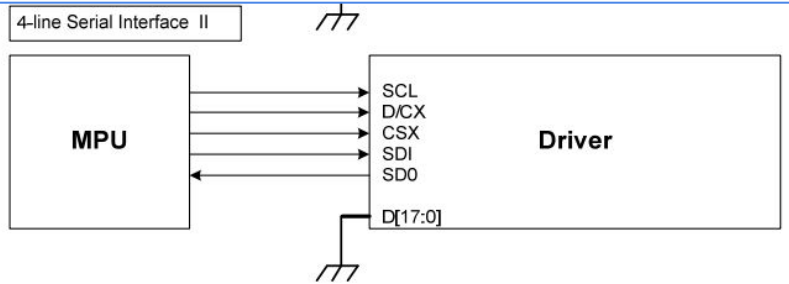


Figure 3.1. Block Diagram of Program Flow

Diagrama Bloques Reproductor MP3 EduCIAA Dual Core



TFT LCD 240x320 Resolution, 262K colors

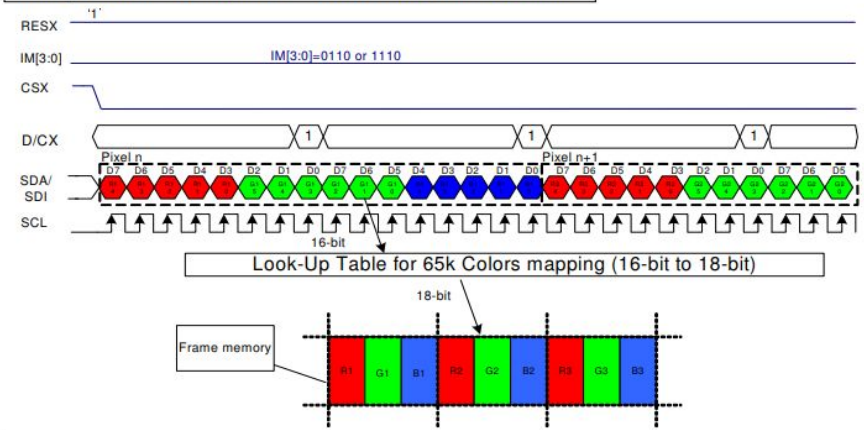


In 4-line serial interface, different display data format is available for two color depths supported by the LCM listed below.

-65k colors, RGB 5, 6, 5 -bits input.

-262k colors, RGB 6, 6, 6 -bits input.

16 bit/pixel color order (R:5-bit, G:6-bit, B:5-bit), 65,536 colors



ILITEK
I Love Innovation

a-Si TFT LCD Single Chip Driver
240RGBx320 Resolution and 262K color

ILI9341

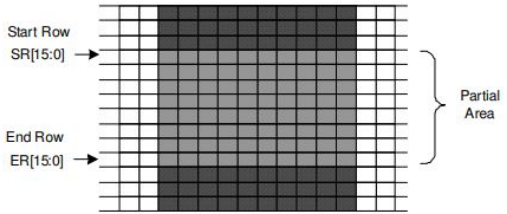
8.2.25. Partial Area (30h)

30h	PLTAR (Partial Area)												
	D/CX	RDX	WRX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	HEX
Command	0	1	↑	XX	0	0	1	1	0	0	0	0	30h
1st Parameter	1	1	↑	XX	SR15	SR14	SR13	SR12	SR11	SR10	SR9	SR8	00
2nd Parameter	1	1	↑	XX	SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0	00
3rd Parameter	1	1	↑	XX	ER15	ER14	ER13	ER12	ER11	ER10	ER9	ER8	01
4th Parameter	1	1	↑	XX	ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0	3F

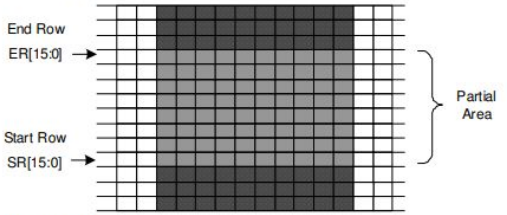
Description

This command defines the partial mode's display area. There are 2 parameters associated with this command, the first defines the Start Row (SR) and the second the End Row (ER), as illustrated in the figures below. SR and ER refer to the Frame Memory Line Pointer.

If End Row>Start Row when MADCTL B4=0:-

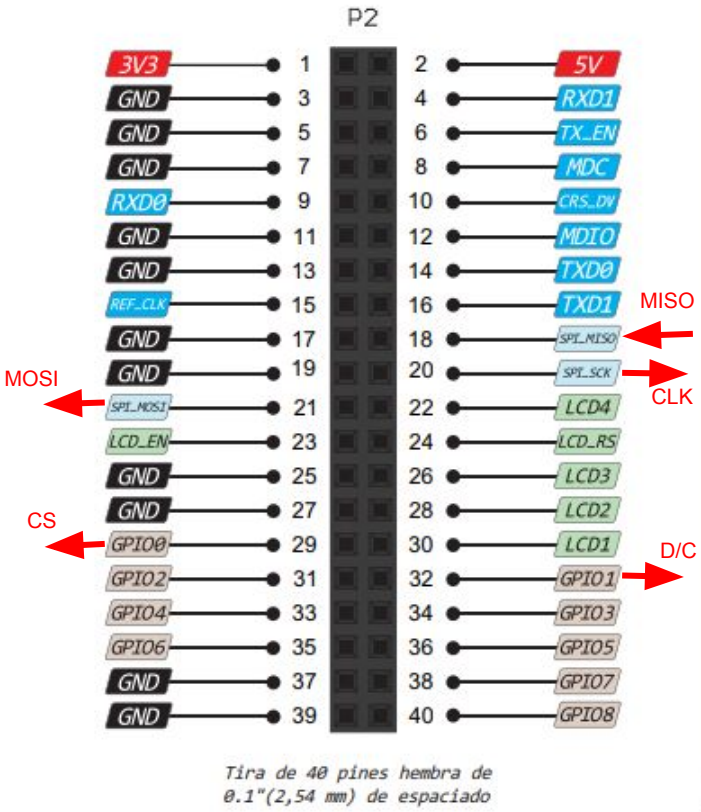


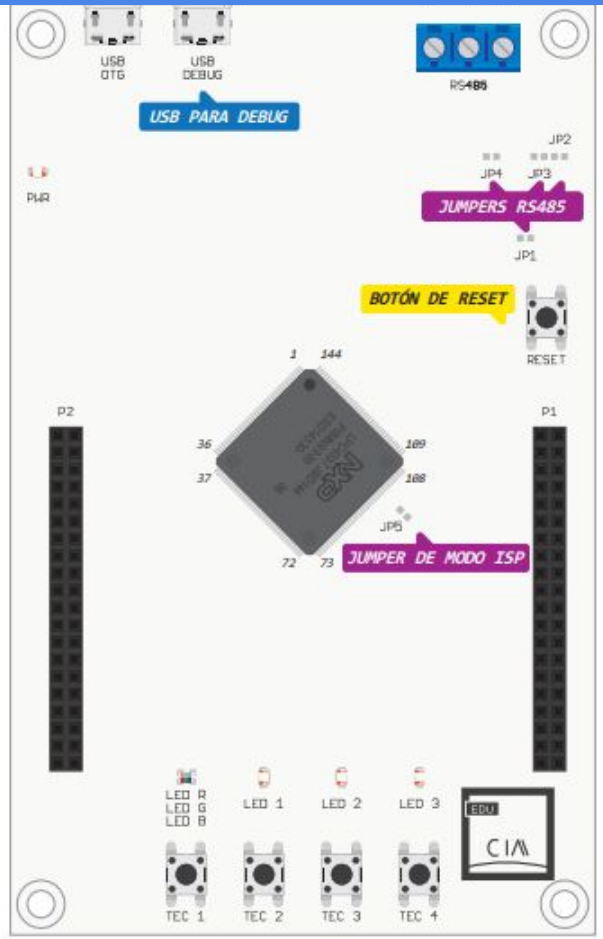
If End Row>Start Row when MADCTL B4=1:-

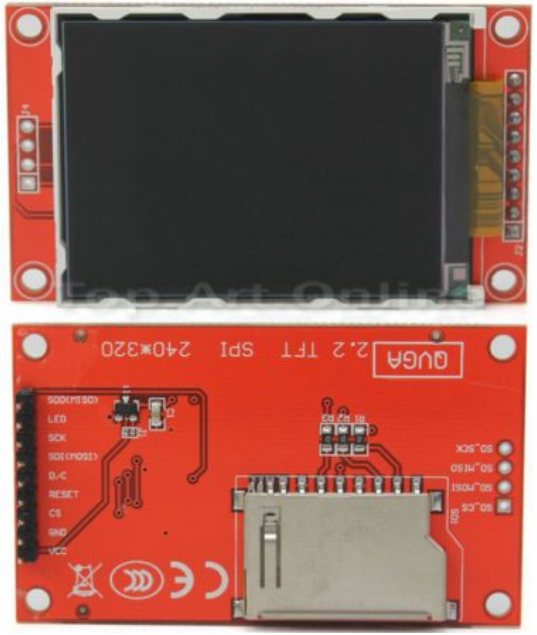


If End Row<Start Row when MADCTL B4=0:-

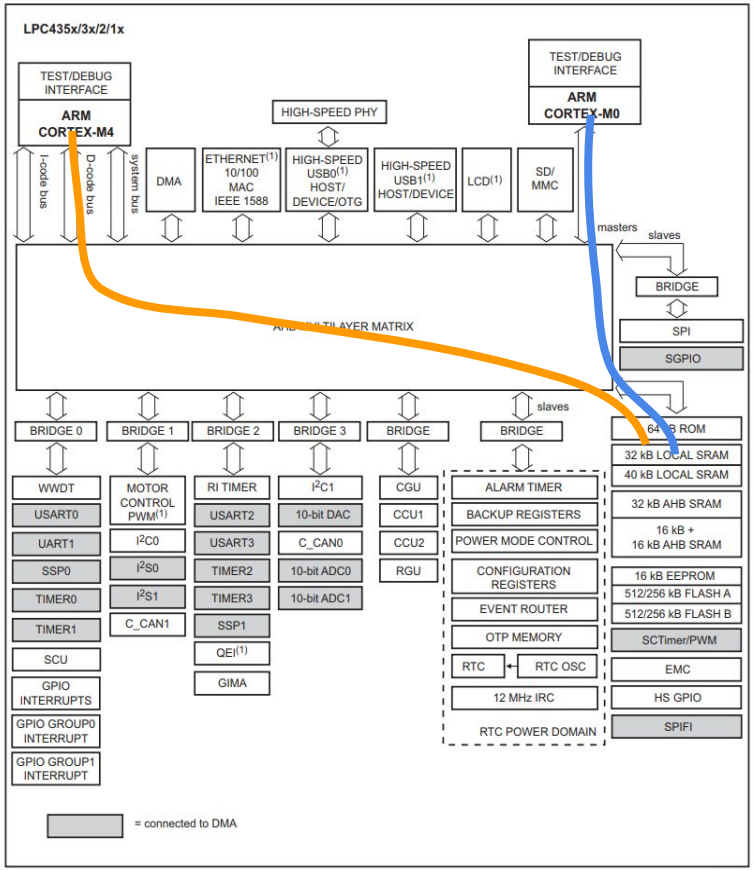
CONEXIÓN EDUCIAA - ILI9341







LPC 4337 RAM



32 kB LOCAL SRAM

40 kB LOCAL SRAM

32 kB AHB SRAM

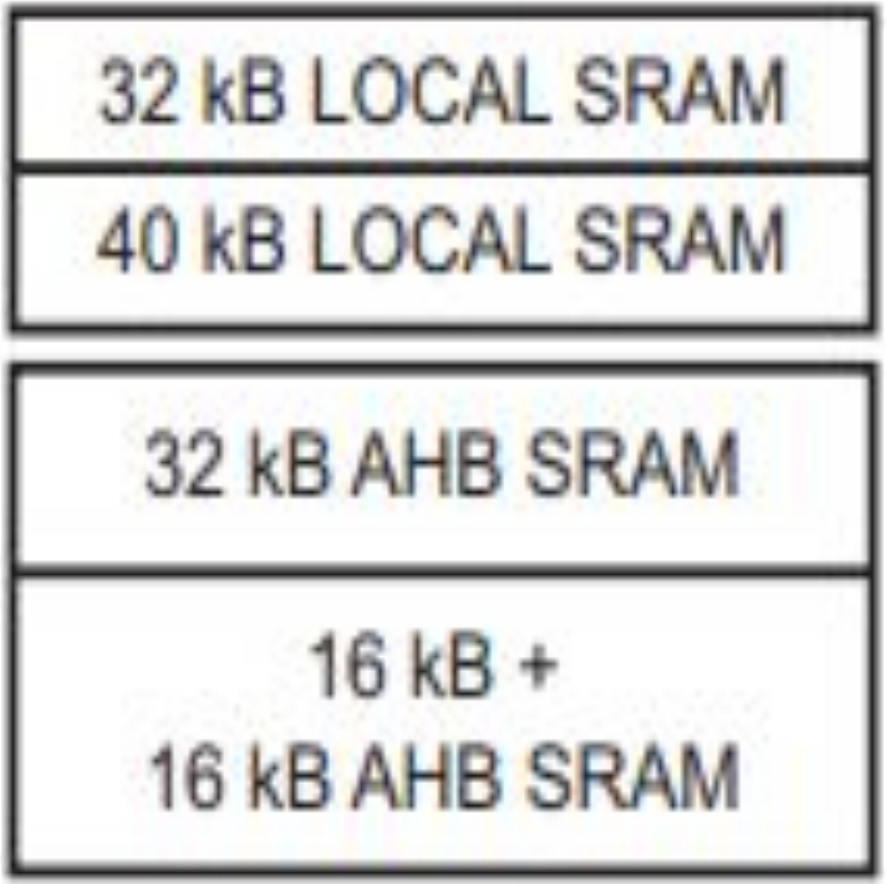
16 kB +
16 kB AHB SRAM

16 kB EEPROM

512/256 kB FLASH A

512/256 kB FLASH B

Structs Secciones RAM EduCIAA



memory.h

```
struct Loc32 {};

struct Loc40 {
        float32_t Input     [SAMPLES];
        float32_t Output  [SAMPLES/2];
        float32_t FFTOutput [SAMPLES/4];
};

struct AHB32{
        struct ipc_sipc;
};

struct AHB16 {
        short outBuf0[2*DECODEBUF_SIZE];
        short outBuf1[2*DECODEBUF_SIZE];
};

struct AHB_ETB16{};
```

IPC Struct

```
struct Loc32 {};

struct Loc40 {
        float32_t Input    [SAMPLES];
        float32_t Output   [SAMPLES/2];
        float32_t FFTOutput[SAMPLES/4];
};

struct AHB32{
        struct ipc_sipc;
};

struct AHB16 {
        short outBuf0[2*DECODEBUF_SIZE];
        short outBuf1[2*DECODEBUF_SIZE];
};

struct AHB_ETB16{};
```

memory.h

```
struct ipc_s {
        int fft_writting;               // offset 0
        unsigned short dummy1;
        int stop_graph;                 // offset 4
        unsigned short dummy2;
        int fft_writting_ack;           // offset 8
        unsigned short dummy3;
        int stop_graph_ack;             // offset 12
        unsigned short dummy;
};
```

IPC

main_m4.c

```
// Play until end of file

  gpioWrite( LED3, ON );
        Enable_DAC = 1;
        ipc->stop_graph = 0;      // inicia dibujo en m0
do
        {
        MP3Play_Frame();
        static int skip = 0;
        if (++skip == 3)
                {
                ipc->fft_writting = 1;
                fft();
                ipc->fft_writting = 0;
                skip=0;
                }
        Revisar_teclas();
  if (tecla>1) break;
  } while (outOfData == 0);
```

main_m0.c

```
while (1)
        {
        while (ipc->stop_graph == 1);          // Dibuja el espectro si se lo permite M4
        /* Set cursor position */
        TM_ILI9341_SetCursorPosition(X0, Y0, X0+ANCHO-1, Y0+ALTO-1);
        /* Set command for GRAM data */
        TM_ILI9341_SendCommand(0x2C);
        Board_SSP_config(16,SSP_CLOCK_CPHA0_CPOL0 ,40000000);
        while (1)
                {
                while(ipc->fft_writting == 1);       // Espero si se está calculando la fft
                int x,y;
                uint16_t color;
                        .
                        .
                        .
```

Timer IRQ

```
timer0Init (Chip_Clock_GetBaseClocktHz (CLK_BASE_MX)/mp3FrameInfo.samprate);
```

timer.c

```
void TIMER0_IRQHandler(){
	int32_t tmp;
	if (Chip_TIMER_MatchPending(LPC_TIMER0, 0))
	{
		static uint16_t indice = 0;
		if (Enable_DAC)
			{
			if (indice >= mp3FrameInfo.outputSamps)
			  {
				tmp = outBuf0[indice-mp3FrameInfo.outputSamps]+32767;
				if (mp3FrameInfo.nChans == 2)                                   // Si es estereo promedio las muestras
					tmp = (tmp + outBuf0[++indice-mp3FrameInfo.outputSamps]+32767)/2;
			  }
			else
			  {
				tmp = outBuf1[indice]+32767;
				if (mp3FrameInfo.nChans == 2)                                   // Si es estereo promedio las muestras
					tmp = (tmp + outBuf1[++indice]+32767)/2;
			  }
			Chip_DAC_UpdateValue(LPC_DAC,(tmp>>6)/2);                           // Mitad para no saturar line-in
			FM_Play (tmp>>9);
			indice++;
			if (indice == mp3FrameInfo.outputSamps*2)
				{
				indice=0;
				getNextFrame = true;
				}
			if (indice == mp3FrameInfo.outputSamps)
				{
				getNextFrame = true;
				}
			}
		else indice = 0;
		Chip_TIMER_ClearMatch(LPC_TIMER0, 0);
	}
}
```

FFT en Core M4 (FPU)

fft.c

```
/* ARM CFFT module */
arm_cfft_radix4_instance_f32 S;

/* Initialize the CFFT/CIFFT module, intFlag = 0, doBitReverse = 1 */
arm_cfft_radix4_init_f32(&S, FFT_SIZE, 0, 1);

/* Process the data through the CFFT/CIFFT module */
arm_cfft_radix4_f32(&S, Input);

/* Process the data through the Complex Magniture Module for calculating the magnitude at each bin */
arm_cmplx_mag_f32(Input, Output, FFT_SIZE);

/* Calculates maxValue and returns corresponding value */
arm_max_f32(Output+30, (FFT_SIZE/2), &maxValue, &maxIndex);


ipc->fft_writting=1;

/* Escalo y copio los resultados pero como int para M0*/
for (i = 0; i < FFT_SIZE/2; i++)
        {
        FFTOutput[i]=(int16_t)(40*(float32_t)((float32_t)Output[i]/(float32_t)maxValue));
        }

ipc->fft_writting=0;
```

LPC 4337 PPL0Audio para FM

```
void FM_Init_PLL()
{
// FM con PLL0AUDIO
// IDIVA usa como fuente PPL1 (204MHz) y lo divide en 4
// PLLAUDIO en modo fraccional y usa como fuente a IDIVA (204/4=51MHz)

        LPC_CGU->IDIV_CTRL[CLK_IDIV_A] =                    // (Table 138) 0x4005 0048 - IDIVA control register
                                        (0x03<<2)|          // divider value = 4
                                        (0x09<<24);         // CLK_SEL = PLL1;

        LPC_CGU->PLL[CGU_AUDIO_PLL].PLL_NP_DIV  =           // (Table 134) 0x4005 0038 - PLL0AUDIO_NP_DIV - divider register
                                        (66)|               // Post-Divider: [6:0]   PDEC Decoded P-divider coefficient value
                                        (0<<12);      // Pre-Divider:  [21:12] NDEC Decoded N-divider coefficient value

        LPC_CGU->PLL0AUDIO_FRAC = (uint32_t)(PLL_FRAC);     // (Table 135) 0x4005 003C bits 21:0 - PLL0AUDIO fractional divider register


        LPC_CGU->PLL[CGU_AUDIO_PLL].PLL_CTRL =              // (Table 132) 0x4005 0030 - PLL0AUDIO control reg
                                        (0<<0)|             // PLL0 power UP
                                        (0<<1)|             // BYPASS
                                        (1<<2)|             // DIRECTI
                                        (0<<3)|             // DIRECTO
                                        (1<<4)|             // CLKEN
                                        (1<<12)|      // PLLFRACT_REQ
                                        (0x0C<<24); // IDIVA (Clock source selection)
```

fm.c

```
void FM_Play (uint8_t modulacion)                   // FM Modulator
{
  LPC_CGU->PLL0AUDIO_FRAC =  (uint32_t)(PLL_FRAC + modulacion);

  LPC_CGU->PLL[CGU_AUDIO_PLL].PLL_CTRL =   // (Table 132) 0x4005 0030 - PLL0A
                                        (0<<0)|             // PLL0 power UP
                                        (0<<1)|             // BYPASS
                                        (1<<2)|             // DIRECTI
                                        (0<<3)|             // DIRECTO
                                        (1<<4)|             // CLKEN
                                        (1<<12)|            // PLLFRACT_REQ
                                        (0x0C<<24);         // IDIVA (Clock source selection)
}
```

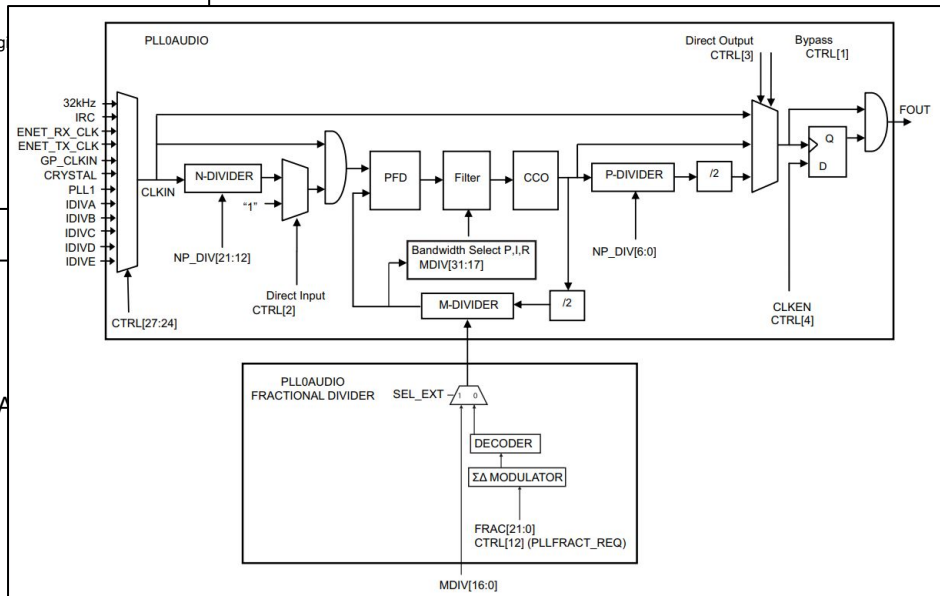


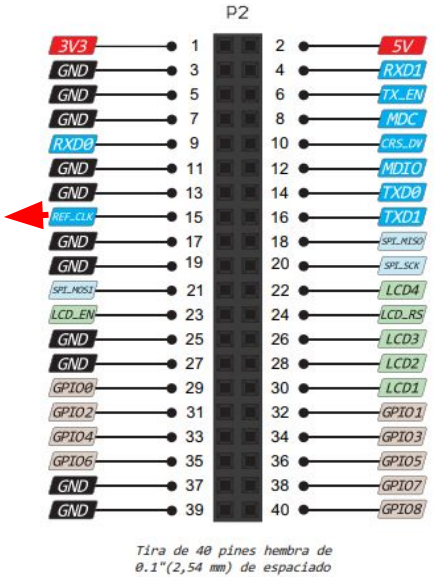
Fig 39. PLL0 with fractional divider

LPC 4337 PPL0Audio salida por Pin

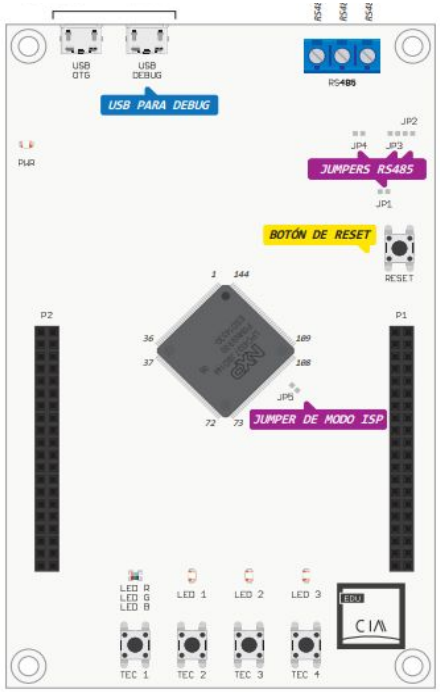
REF_CLK — 68 | P1_19 | ENET_TX_CLK | SSP1_SCK | R | R | CLKOUT | R | I2S0_RX_MCLK | I2S1_TX_SCK

```
/* Ruteo PLLAUDIO al pin REF CLK de la EduCIAA*/
	Chip_Clock_SetBaseClock(CLK_BASE_OUT, CLKIN_AUDIOPLL, true, false);
	// (Table 146) 0x400500AC bits 28:24 - BASE_OUT_CLK source selection:
CLK_SEL = PLL0AUDIO
	Chip_SCU_PinMuxSet(0x1, 19, (SCU_PINIO_FAST | SCU_MODE_FUNC4));
	// (Table 191) 0x400860CC bits 2:0 - PIN1.19 MODE: Function 4 = CLKOUT
```

fm.c



Tira de 40 pines hembra de 0.1"(2,54 mm) de espaciado



Readme

Compilación

Para compilar y grabar el M4:

make

make download

Para compilar y grabar el M0

make TARGET=lpc4337_m0

make TARGET=lpc4337_m0 download

Readme

Utilización

Funciona sobre la EDU-CIAA-NXP. Conectar un pendrive al USB conteniendo archivos mp3 en el directorio raíz.

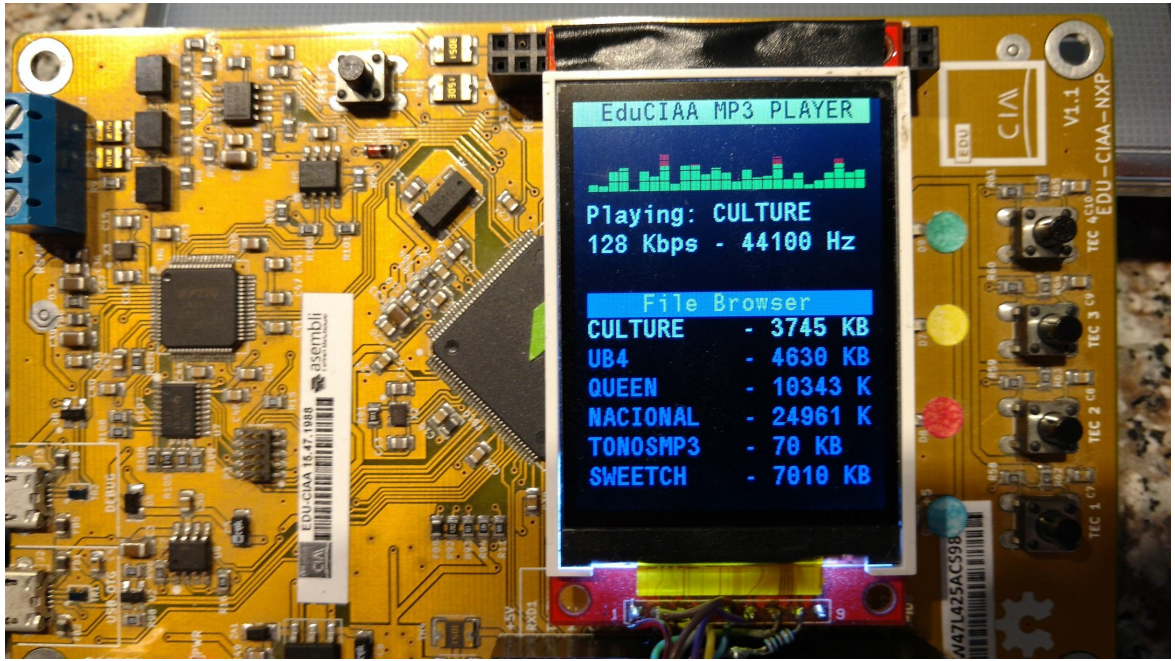
TEC2 y TEC3 para seleccionar archivo el “File Browser” y TEC1 para Play/Stop.

Las conexiones al TFT LCD ILI9341 son:

SD/MicroSD Card Reader --> EDU-CIAA-NXP

```
 + --> +3.3V
 CS --> GPIO0
 D/C --> GPIO1
 DI --> SPI_MOSI
CLK --> SPI_SCK
 DO --> SPI_MISO
  G --> GND
```

Gracias por su atención!



https://github.com/telmomoya/educiaa-cese