**Arduino IDEのFM3 USB Host Driverの対応**

——IAR SYSTEMSの MB9BF618T-SK ボードに基づき

**アジェンダ**

1． Interface誌付録FM3基板の対応

*※ 以下の対応はIAR SYSTEMSの MB9BF618T-SK ボードに基づいている。*

2．LEDの対応

3．UARTの対応

4．USB Host Driverの対応

5．USB Host Applicationの対応

6．工程のビルドおよび検証

# 1．Interface誌付録FM3基板の対応

(1) arduino-1.5.4-r2-windows.exeをD:\Arduinoにインストールする

(2) 《Arduino IDEのFM3対応～Interface誌付録FM3基板でＬチカまで～》（arduino\_yistudio3.pdf）手順書に記載されている手順に従って対応する

# 2．LEDの対応

**(1)** D:\ Arduino\hardware\arduino\sam\system\libsam\source\led.cを変更する

#include "mcu.h"

/// @cond 0

/\*\*INDENT-OFF\*\*/

#ifdef \_\_cplusplus

extern "C" {

#endif

/\*\*INDENT-ON\*\*/

/// @endcond

/\* LED pin definition \*/

#define LED\_USER (1UL<<4)

#define LED\_MASK (LED\_USER)

#define LED\_PFR FM3\_GPIO->PFRF

#define LED\_DDR FM3\_GPIO->DDRF

#define LED\_PDOR FM3\_GPIO->PDORF

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* LED port initialize

\*/

void led\_init( void )

{

/\* Select GPIO function \*/

LED\_PFR &= ~LED\_MASK;

/\* Set pin to turn off led \*/

LED\_PDOR |= LED\_MASK;

/\* Make led pin output \*/

LED\_DDR |= LED\_MASK;

}

/\*

\* bit0 : 0 : LED OFF

\* 1 : LED ON

\*/

void led\_set( unsigned int led )

{

if( led & 0x00000001 ) {

/\* LED ON \*/

LED\_PDOR &= ~LED\_MASK;

} else {

/\* LED OFF \*/

LED\_PDOR |= LED\_MASK;

}

}

**(2)** D:\ Arduino\hardware\arduino\sam\system\libsam\build\_gcc\Makefileを変更する

(line108)

.PHONY: clean

clean:

@echo ------------------------------------------------------------------------------------

@echo --- Cleaning sam3s4c release and debug

@$(MAKE) CHIP=\_\_SAM3S4C\_\_ $(SUBMAKE\_OPTIONS) -f sam3.mk $@

@$(MAKE) CHIP=\_\_SAM3S4C\_\_ DEBUG=1 $(SUBMAKE\_OPTIONS) -f sam3.mk $@

@echo --- Cleaning sam3u4e release and debug

@$(MAKE) CHIP=\_\_SAM3U4E\_\_ $(SUBMAKE\_OPTIONS) OUTPUT\_BIN=../../../variants/arduino\_due\_x -f sam3.mk $@

@$(MAKE) CHIP=\_\_SAM3U4E\_\_ DEBUG=1 $(SUBMAKE\_OPTIONS) OUTPUT\_BIN=../../../variants/arduino\_due\_x -f sam3.mk $@

@echo --- Cleaning sam3x8e release and debug

@$(MAKE) CHIP=\_\_SAM3X8E\_\_ $(SUBMAKE\_OPTIONS) OUTPUT\_BIN=../../../variants/arduino\_due\_x -f sam3.mk $@

@$(MAKE) CHIP=\_\_SAM3X8E\_\_ DEBUG=1 $(SUBMAKE\_OPTIONS) OUTPUT\_BIN=../../../variants/arduino\_due\_x -f sam3.mk $@

**(3)** libsamライブラリファイルをビルドする

ＤＯＳ窓（コマンドプロンプト）を開いて、以下の操作を実行する。

ディレクトリ名は各自がarduinoを展開したディレクトリ名へ変更する必要がある。

cd D:\Arduino\hardware\arduino\sam\system\libsam\build\_gcc

cs-make clean

cs-make -f Makefile ARM\_GCC\_TOOLCHAIN=D:\Arduino\hardware\tools\g++\_arm\_none\_eabi\bin MAKE=cs-make arduino\_due\_x

ビルドで生成されたオブジェクトファイル（ ～.o ）が置かれるディレクトリ

D:\Arduino\hardware\arduino\sam\system\libsam\build\_gcc\release\_sam3x8e

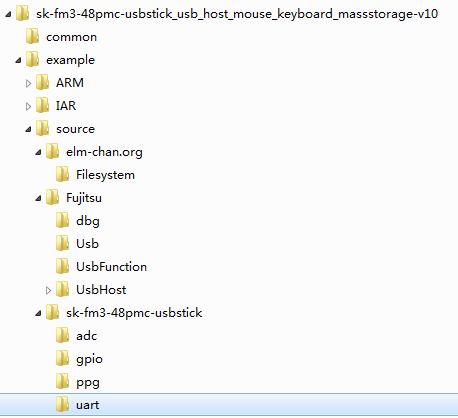
ビルドで生成されたライブラリ

D:\Arduino\hardware\arduino\sam\variants\arduino\_due\_x\libsam\_sam3x8e\_gcc\_rel.a

# 3．UARTの対応

UARTソースはSK-FM3-48PMC-USBSTICK光ディスクの以下のディレクトリより取得されたものである：

CD\Examples\sk-fm3-48pmc-usbstick\_usb\_host\_mouse\_keyboard\_massstorage-v10



**(1)** sk-fm3-48pmc-usbstick\uart\uart.hを以下のパスの配下にコピーする：

D:\Arduino\hardware\arduino\sam\system\libsam\include

また、以下の変更を行う：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef INCLUDE\_FROM\_CHIP\_H

#include "mcu.h"

#else

#include "sysdef\_depend.h"

#endif

#include "../source/spansion/base\_type.h"

#include <string.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Global pre-processor symbols/macros ('#define') \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define USE\_PRINTF 0

#if (USE\_PRINTF == 1)

#include <stdio.h>

#endif

#define HEAP\_SIZE 16

#ifndef CLOCK\_FREQ

#define CLOCK\_FREQ 40000000UL

#endif

#define UART\_BAUDRATE(x) ((SystemCoreClock / 2 + (x / 2)) / x - 1)

**(2)** sk-fm3-48pmc-usbstick\uart\uart.cを以下のパスの配下にコピーする：

D:\Arduino\hardware\arduino\sam\system\libsam\source

また、以下の変更を行う：

FM3\_MFS03\_UART\_TypeDef\* UART\_BASE[] = {

((FM3\_MFS03\_UART\_TypeDef \*)FM3\_MFS0\_UART\_BASE),

((FM3\_MFS03\_UART\_TypeDef \*)FM3\_MFS1\_UART\_BASE),

((FM3\_MFS03\_UART\_TypeDef \*)FM3\_MFS2\_UART\_BASE),

((FM3\_MFS03\_UART\_TypeDef \*)FM3\_MFS3\_UART\_BASE),

((FM3\_MFS03\_UART\_TypeDef \*)FM3\_MFS4\_UART\_BASE),

((FM3\_MFS03\_UART\_TypeDef \*)FM3\_MFS5\_UART\_BASE),

((FM3\_MFS03\_UART\_TypeDef \*)FM3\_MFS6\_UART\_BASE),

((FM3\_MFS03\_UART\_TypeDef \*)FM3\_MFS7\_UART\_BASE)};

…

…

…

boolean\_t Uart\_Init(uint8\_t u8Uart, uint16\_t Baudrate)

{

UART\_BASE[u8Uart]->SCR = 0x80;

UART\_BASE[u8Uart]->SMR = 0x01;

UART\_BASE[u8Uart]->SSR = 0x00;

UART\_BASE[u8Uart]->ESCR = 0x00;

UART\_BASE[u8Uart]->BGR = Baudrate; //(uint16\_t)BGRTemp;

UART\_BASE[u8Uart]->SCR = 0x03;

if (u8Uart == 0)

{

FM3\_GPIO->ADE &= ~0x80000000; // Disable AN31

FM3\_GPIO->PFR2 |= 0x0006; // SIN0\_0, SOT0\_0, SCK0\_0

FM3\_GPIO->EPFR07 |= 0x00000050; // SIN0\_0, SOT0\_0, SCK0\_0

}

u8DefaultUart = u8Uart;

return TRUE;

}

**(3)** D:\Arduino\hardware\arduino\sam\system\libsam\source\ sysdef\_depend.hファイルを

D:\Arduino\hardware\arduino\sam\system\libsam\include\パスの配下に移動する。

また、以下の変更を行う：

#ifndef \_SYSDEF\_MB9BF618T\_H\_

#define \_SYSDEF\_MB9BF618T\_H\_

#ifdef \_\_cplusplus

extern "C" {

#endif

#define FM3\_PERIPH\_BASE (0x40000000UL)

#define FM3\_GPIO\_BASE (FM3\_PERIPH\_BASE + 0x33000UL)

#define MFS0\_UART\_BASE (FM3\_PERIPH\_BASE + 0x38000UL)

#define MFS1\_UART\_BASE (FM3\_PERIPH\_BASE + 0x38100UL)

#define MFS2\_UART\_BASE (FM3\_PERIPH\_BASE + 0x38200UL)

#define MFS3\_UART\_BASE (FM3\_PERIPH\_BASE + 0x38300UL)

#define \_\_IO volatile

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* GPIO\_MODULE General purpose I/O registers

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef struct

{

\_\_IO uint32\_t PFR0;

\_\_IO uint32\_t PFR1;

\_\_IO uint32\_t PFR2;

\_\_IO uint32\_t PFR3;

\_\_IO uint32\_t PFR4;

\_\_IO uint32\_t PFR5;

\_\_IO uint32\_t PFR6;

\_\_IO uint32\_t PFR7;

\_\_IO uint32\_t PFR8;

\_\_IO uint32\_t PFR9;

\_\_IO uint32\_t PFRA;

\_\_IO uint32\_t PFRB;

\_\_IO uint32\_t PFRC;

\_\_IO uint32\_t PFRD;

\_\_IO uint32\_t PFRE;

\_\_IO uint32\_t PFRF;

uint8\_t RESERVED0[192];

\_\_IO uint32\_t PCR0;

\_\_IO uint32\_t PCR1;

\_\_IO uint32\_t PCR2;

\_\_IO uint32\_t PCR3;

\_\_IO uint32\_t PCR4;

\_\_IO uint32\_t PCR5;

\_\_IO uint32\_t PCR6;

\_\_IO uint32\_t PCR7;

\_\_IO uint32\_t PCR8;

\_\_IO uint32\_t PCR9;

\_\_IO uint32\_t PCRA;

\_\_IO uint32\_t PCRB;

\_\_IO uint32\_t PCRC;

\_\_IO uint32\_t PCRD;

\_\_IO uint32\_t PCRE;

\_\_IO uint32\_t PCRF;

uint8\_t RESERVED1[192];

\_\_IO uint32\_t DDR0;

\_\_IO uint32\_t DDR1;

\_\_IO uint32\_t DDR2;

\_\_IO uint32\_t DDR3;

\_\_IO uint32\_t DDR4;

\_\_IO uint32\_t DDR5;

\_\_IO uint32\_t DDR6;

\_\_IO uint32\_t DDR7;

\_\_IO uint32\_t DDR8;

\_\_IO uint32\_t DDR9;

\_\_IO uint32\_t DDRA;

\_\_IO uint32\_t DDRB;

\_\_IO uint32\_t DDRC;

\_\_IO uint32\_t DDRD;

\_\_IO uint32\_t DDRE;

\_\_IO uint32\_t DDRF;

uint8\_t RESERVED2[192];

\_\_IO uint32\_t PDIR0;

\_\_IO uint32\_t PDIR1;

\_\_IO uint32\_t PDIR2;

\_\_IO uint32\_t PDIR3;

\_\_IO uint32\_t PDIR4;

\_\_IO uint32\_t PDIR5;

\_\_IO uint32\_t PDIR6;

\_\_IO uint32\_t PDIR7;

\_\_IO uint32\_t PDIR8;

\_\_IO uint32\_t PDIR9;

\_\_IO uint32\_t PDIRA;

\_\_IO uint32\_t PDIRB;

\_\_IO uint32\_t PDIRC;

\_\_IO uint32\_t PDIRD;

\_\_IO uint32\_t PDIRE;

\_\_IO uint32\_t PDIRF;

uint8\_t RESERVED3[192];

\_\_IO uint32\_t PDOR0;

\_\_IO uint32\_t PDOR1;

\_\_IO uint32\_t PDOR2;

\_\_IO uint32\_t PDOR3;

\_\_IO uint32\_t PDOR4;

\_\_IO uint32\_t PDOR5;

\_\_IO uint32\_t PDOR6;

\_\_IO uint32\_t PDOR7;

\_\_IO uint32\_t PDOR8;

\_\_IO uint32\_t PDOR9;

\_\_IO uint32\_t PDORA;

\_\_IO uint32\_t PDORB;

\_\_IO uint32\_t PDORC;

\_\_IO uint32\_t PDORD;

\_\_IO uint32\_t PDORE;

\_\_IO uint32\_t PDORF;

uint8\_t RESERVED4[192];

\_\_IO uint32\_t ADE;

uint8\_t RESERVED5[124];

\_\_IO uint32\_t SPSR;

uint8\_t RESERVED6[124];

\_\_IO uint32\_t EPFR00;

\_\_IO uint32\_t EPFR01;

\_\_IO uint32\_t EPFR02;

\_\_IO uint32\_t EPFR03;

\_\_IO uint32\_t EPFR04;

\_\_IO uint32\_t EPFR05;

\_\_IO uint32\_t EPFR06;

\_\_IO uint32\_t EPFR07;

\_\_IO uint32\_t EPFR08;

\_\_IO uint32\_t EPFR09;

\_\_IO uint32\_t EPFR10;

\_\_IO uint32\_t EPFR11;

\_\_IO uint32\_t EPFR12;

\_\_IO uint32\_t EPFR13;

\_\_IO uint32\_t EPFR14;

\_\_IO uint32\_t EPFR15;

uint8\_t RESERVED7[192];

\_\_IO uint32\_t PZR0;

\_\_IO uint32\_t PZR1;

\_\_IO uint32\_t PZR2;

\_\_IO uint32\_t PZR3;

\_\_IO uint32\_t PZR4;

\_\_IO uint32\_t PZR5;

\_\_IO uint32\_t PZR6;

\_\_IO uint32\_t PZR7;

\_\_IO uint32\_t PZR8;

\_\_IO uint32\_t PZR9;

\_\_IO uint32\_t PZRA;

\_\_IO uint32\_t PZRB;

\_\_IO uint32\_t PZRC;

\_\_IO uint32\_t PZRD;

\_\_IO uint32\_t PZRE;

\_\_IO uint32\_t PZRF;

}FM3\_GPIO\_TypeDef;

#define FM3\_GPIO ((FM3\_GPIO\_TypeDef \*)FM3\_GPIO\_BASE)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MFS\_UART\_MODULE UART asynchronous channel registers

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef struct

{

\_\_IO uint8\_t SMR;

\_\_IO uint8\_t SCR;

uint8\_t RESERVED0[2];

\_\_IO uint8\_t ESCR;

\_\_IO uint8\_t SSR;

uint8\_t RESERVED1[2];

union {

\_\_IO uint16\_t RDR;

\_\_IO uint16\_t TDR;

};

uint8\_t RESERVED2[2];

union {

\_\_IO uint16\_t BGR;

struct {

\_\_IO uint8\_t BGR0;

\_\_IO uint8\_t BGR1;

};

};

}FM3\_MFS03\_UART\_TypeDef;

#define UART0 ((FM3\_MFS03\_UART\_TypeDef \*)MFS0\_UART\_BASE)

#define UART1 ((FM3\_MFS03\_UART\_TypeDef \*)MFS1\_UART\_BASE)

#define UART2 ((FM3\_MFS03\_UART\_TypeDef \*)MFS2\_UART\_BASE)

#define UART3 ((FM3\_MFS03\_UART\_TypeDef \*)MFS3\_UART\_BASE)

/\*

\* UART Register Bit Definition

\*/

#define SMR\_SOE 0x01U

#define SMR\_BDS 0x04U

#define SMR\_SBL 0x08U

#define SMR\_WUCR 0x10U

#define SMR\_MD\_UART 0x00U

#define SMR\_MD\_UART\_MP 0x20U

#define SMR\_MD\_SIO 0x40U

#define SMR\_MD\_LIN 0x60U

#define SMR\_MD\_I2C 0x80U

#define SCR\_TXE 0x01U

#define SCR\_RXE 0x02U

#define SCR\_TBIE 0x04U

#define SCR\_TIE 0x08U

#define SCR\_RIE 0x10U

#define SCR\_UPGL 0x80U

#define SSR\_TBI 0x01U

#define SSR\_TDRE 0x02U

#define SSR\_RDRF 0x04U

#define SSR\_ORE 0x08U

#define SSR\_FRE 0x10U

#define SSR\_PE 0x20U

#define SSR\_REC 0x80U

#define ESCR\_P 0x08U

#define ESCR\_PEN 0x10U

#define ESCR\_INV 0x20U

#define ESCR\_ESBL 0x40U

#define ESCR\_FLWEN 0x80U

#define ESCR\_DATABITS\_8 0x00U

#define ESCR\_DATABITS\_5 0x01U

#define ESCR\_DATABITS\_6 0x02U

#define ESCR\_DATABITS\_7 0x03U

#define ESCR\_DATABITS\_9 0x04U

#ifdef \_\_cplusplus

}

#endif

#endif /\* \_SYSDEF\_MB9BF618T\_H\_ \*/

**(4)** D:\Arduino\hardware\arduino\sam\cores\arduino\UARTClass.cppを変更する

void UARTClass::begin( const uint32\_t dwBaudRate )

{

#if 0

// Configure PMC

pmc\_enable\_periph\_clk( \_dwId ) ;

// Disable PDC channel

\_pUart->UART\_PTCR = UART\_PTCR\_RXTDIS | UART\_PTCR\_TXTDIS ;

// Reset and disable receiver and transmitter

\_pUart->UART\_CR = UART\_CR\_RSTRX | UART\_CR\_RSTTX | UART\_CR\_RXDIS | UART\_CR\_TXDIS ;

// Configure mode

\_pUart->UART\_MR = UART\_MR\_PAR\_NO | UART\_MR\_CHMODE\_NORMAL ;

// Configure baudrate (asynchronous, no oversampling)

\_pUart->UART\_BRGR = (SystemCoreClock / dwBaudRate) >> 4 ;

// Configure interrupts

\_pUart->UART\_IDR = 0xFFFFFFFF;

\_pUart->UART\_IER = UART\_IER\_RXRDY | UART\_IER\_OVRE | UART\_IER\_FRAME;

// Enable UART interrupt in NVIC

NVIC\_EnableIRQ(\_dwIrq);

// Enable receiver and transmitter

\_pUart->UART\_CR = UART\_CR\_RXEN | UART\_CR\_TXEN ;

#endif

/\* FM3 MFS0 initialization \*/

FM3\_GPIO->ADE &= ~0x80000000; // Disable AN31

FM3\_GPIO->PFR2 |= 0x0006; // SIN0\_0, SOT0\_0, SCK0\_0

FM3\_GPIO->EPFR07 |= 0x00000050; // SIN0\_0, SOT0\_0, SCK0\_0

UART0->SMR = (SMR\_MD\_UART | SMR\_SOE);

UART0->SCR = 0x00;

UART0->BGR = (SystemCoreClock / 2 + (dwBaudRate / 2)) / dwBaudRate - 1;

UART0->ESCR = 0x0000;

NVIC\_EnableIRQ(MFS0RX\_IRQn); // Enable receive interupt

UART0->SCR |= (SCR\_RXE | SCR\_TXE | SCR\_RIE);

}

void UARTClass::end( void )

{

// clear any received data

\_rx\_buffer->\_iHead = \_rx\_buffer->\_iTail ;

#if 0

// Disable UART interrupt in NVIC

NVIC\_DisableIRQ( \_dwIrq ) ;

// Wait for any outstanding data to be sent

flush();

pmc\_disable\_periph\_clk( \_dwId ) ;

#endif

NVIC\_DisableIRQ( MFS0RX\_IRQn ) ;

flush();

}

…

…

…

void UARTClass::flush( void )

{

// Wait for transmission to complete

#if 0

while ((\_pUart->UART\_SR & UART\_SR\_TXRDY) != UART\_SR\_TXRDY)

;

#endif

while((UART0->SSR & SSR\_TDRE) == 0);

}

size\_t UARTClass::write( const uint8\_t uc\_data )

{

// Check if the transmitter is ready

#if 0

while ((\_pUart->UART\_SR & UART\_SR\_TXRDY) != UART\_SR\_TXRDY)

;

// Send character

\_pUart->UART\_THR = uc\_data;

#endif

while((UART0->SSR & SSR\_TDRE) == 0);

UART0->TDR = uc\_data;

return 1;

}

void UARTClass::IrqHandler( void )

{

#if 0

uint32\_t status = \_pUart->UART\_SR;

// Did we receive data ?

if ((status & UART\_SR\_RXRDY) == UART\_SR\_RXRDY)

\_rx\_buffer->store\_char(\_pUart->UART\_RHR);

// Acknowledge errors

if ((status & UART\_SR\_OVRE) == UART\_SR\_OVRE ||

(status & UART\_SR\_FRAME) == UART\_SR\_FRAME)

{

// TODO: error reporting outside ISR

\_pUart->UART\_CR |= UART\_CR\_RSTSTA;

}

#endif

if(UART0->SSR & SSR\_RDRF)

\_rx\_buffer->store\_char(UART0->RDR);

}

**(5)** D:\Arduino\hardware\arduino\sam\system\CMSIS\Device\ATMEL\sam3xa\include\sam3x8e.hを変更する

(line77)

SUPC\_IRQn = 0, /\*\*< 0 SAM3X8E Supply Controller (SUPC) \*/

RSTC\_IRQn = 1, /\*\*< 1 SAM3X8E Reset Controller (RSTC) \*/

RTC\_IRQn = 2, /\*\*< 2 SAM3X8E Real Time Clock (RTC) \*/

RTT\_IRQn = 3, /\*\*< 3 SAM3X8E Real Time Timer (RTT) \*/

WDT\_IRQn = 4, /\*\*< 4 SAM3X8E Watchdog Timer (WDT) \*/

PMC\_IRQn = 5, /\*\*< 5 SAM3X8E Power Management Controller (PMC) \*/

EFC0\_IRQn = 6, /\*\*< 6 SAM3X8E Enhanced Flash Controller 0 (EFC0) \*/

#if 0

EFC1\_IRQn = 7, /\*\*< 7 SAM3X8E Enhanced Flash Controller 1 (EFC1) \*/

#endif

MFS0RX\_IRQn = 7, /\*\*< 7 FM3 MFS0 Receive Interrupt (UART0) \*/

UART\_IRQn = 8, /\*\*< 8 SAM3X8E Universal Asynchronous Receiver Transceiver (UART) \*/

SMC\_IRQn = 9, /\*\*< 9 SAM3X8E Static Memory Controller (SMC) \*/

PIOA\_IRQn = 11, /\*\*< 11 SAM3X8E Parallel I/O Controller A, (PIOA) \*/

PIOB\_IRQn = 12, /\*\*< 12 SAM3X8E Parallel I/O Controller B (PIOB) \*/

**(6)** D:\Arduino\hardware\arduino\sam\system\CMSIS\Device\ATMEL\sam3xa\source\gcc\startup\_sam3xa.cを変更する

(line30)

/\* Initialize segments \*/

extern uint32\_t \_sfixed;

extern uint32\_t \_efixed;

extern uint32\_t \_etext;

extern uint32\_t \_srelocate;

extern uint32\_t \_erelocate;

extern uint32\_t \_szero;

extern uint32\_t \_ezero;

extern uint32\_t \_sstack;

extern uint32\_t \_estack;

extern void MFS0RX\_IRQHandler(void);

…

…

…

/\* Configurable interrupts \*/

(void\*) SUPC\_Handler, /\* 0 Supply Controller \*/

(void\*) RSTC\_Handler, /\* 1 Reset Controller \*/

(void\*) RTC\_Handler, /\* 2 Real Time Clock \*/

(void\*) RTT\_Handler, /\* 3 Real Time Timer \*/

(void\*) WDT\_Handler, /\* 4 Watchdog Timer \*/

(void\*) PMC\_Handler, /\* 5 PMC \*/

(void\*) EFC0\_Handler, /\* 6 EFC 0 \*/

#if 0

(void\*) EFC1\_Handler, /\* 7 EFC 1 \*/

#endif

(void\*) MFS0RX\_IRQHandler, /\* 7 FM3 MFS0 Receive (UART0) \*/

(void\*) UART\_Handler, /\* 8 UART \*/

**(7)** D:\Arduino\hardware\arduino\sam\variants\arduino\_due\_x\variant.cppを変更する

(line287)

extern void MFS0RX\_IRQHandler(void);

#ifdef \_\_cplusplus

}

#endif

/\*

\* UART objects

\*/

RingBuffer rx\_buffer1;

UARTClass Serial(UART, MFS0RX\_IRQn, ID\_UART, &rx\_buffer1);

void MFS0RX\_IRQHandler(void)

{

Serial.IrqHandler();

}

#if 0

UARTClass Serial(UART, UART\_IRQn, ID\_UART, &rx\_buffer1);

// IT handlers

void UART\_Handler(void)

{

Serial.IrqHandler();

}

// ----------------------------------------------------------------------------

/\*

\* USART objects

\*/

RingBuffer rx\_buffer2;

RingBuffer rx\_buffer3;

RingBuffer rx\_buffer4;

USARTClass Serial1(USART0, USART0\_IRQn, ID\_USART0, &rx\_buffer2);

USARTClass Serial2(USART1, USART1\_IRQn, ID\_USART1, &rx\_buffer3);

USARTClass Serial3(USART3, USART3\_IRQn, ID\_USART3, &rx\_buffer4);

// IT handlers

void USART0\_Handler(void)

{

Serial1.IrqHandler();

}

void USART1\_Handler(void)

{

Serial2.IrqHandler();

}

void USART3\_Handler(void)

{

Serial3.IrqHandler();

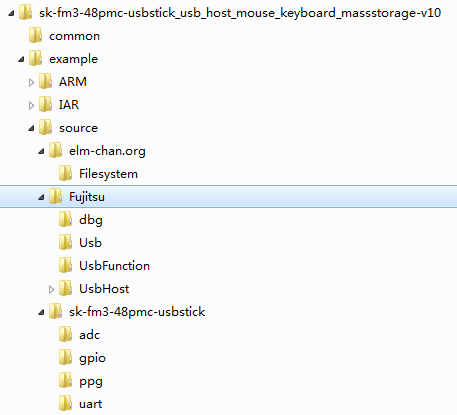
}

#endif

# 4．USB Host Driverの対応

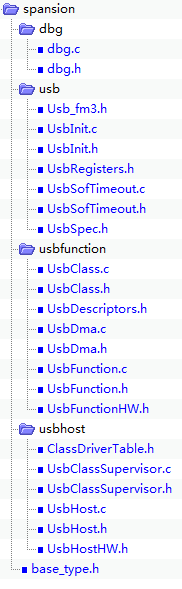
USB Host DriverのソースはSK-FM3-48PMC-USBSTICK光ディスクの以下のディレクトリより取得されたものである：CD\Examples\sk-fm3-48pmc-usbstick\_usb\_host\_mouse\_keyboard\_massstorage-v10

D:\Arduino\hardware\arduino\sam\system\libsam\sourceパス配下にspansionフォルダを新規作成する。ディレクトリ構成を下図に示す。ファイルは上記のフォルダより取得されたものである。



spansion\base\_type.hはexample\source\base\_type.hより取得された。

Arduino固有のファイルにusb\Usb.hと同名するファイルがすでに存在しているため、それをusb\Usb\_fm3.hにリネームする。



**(1)** spansion\base\_type.hを変更する

#ifndef \_\_BASE\_TYPE\_H\_\_

#define \_\_BASE\_TYPE\_H\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stddef.h>

#include "stdint.h"

//#include "mcu.h"

**(2)** spansion\usb\Usb\_fm3.hを変更する

#ifndef \_\_USB\_H\_\_

#define \_\_USB\_H\_\_

#define USE\_USB\_HOST 1

#define USE\_USB\_FUNCTION 0

#define ENABLE\_USB\_HOST\_FUNCION\_SWITCHING 0

#if (USE\_USB\_HOST == 1)

#include "../usbhost/UsbHost.h"

#endif

**(3)** spansion\usb\UsbInit.c を変更する(line88)

#if IS\_USBMCU\_FM3

#define UsbInit() UsbInit\_FM3()

void UsbInit\_FM3(void)

{

volatile unsigned char dummy;

#ifdef USE\_USB0

NVIC\_ClearPendingIRQ(USB0F\_IRQn);

NVIC\_EnableIRQ(USB0F\_IRQn);

NVIC\_SetPriority(USB0F\_IRQn,1);

NVIC\_ClearPendingIRQ(USB0F\_USB0H\_IRQn);

NVIC\_EnableIRQ(USB0F\_USB0H\_IRQn);

NVIC\_SetPriority(USB0F\_USB0H\_IRQn,1);

#else

//NVIC\_ClearPendingIRQ(USBF\_IRQn);

//NVIC\_EnableIRQ(USBF\_IRQn);

//NVIC\_SetPriority(USBF\_IRQn,1);

NVIC\_ClearPendingIRQ(USBF\_USBH\_IRQn);

NVIC\_EnableIRQ(USBF\_USBH\_IRQn);

NVIC\_SetPriority(USBF\_USBH\_IRQn,1);

#endif

/\* USB \*/

bFM3\_GPIO\_SPSR\_USB1C = 1; // USB1C = 1 (use USB pins), SUBXC = 0 (Do not use subclock)

FM3\_GPIO->EPFR00 |= 0x00002000; /\* USB1PE=1 \*/

/\* USB Pull-up : P61 as UHCONX \*/

//FM3\_GPIO->PFR6 = 0x0002;

/\* set USB clock \*/

#if ((USBMCUTYPE == USBMCU\_MB9B500) || (USBMCUTYPE == USBMCU\_MB9A310))

FM3\_USBCLK->UCCR = 0x00;

FM3\_USBCLK->USBEN = 0x04; /\* USBEN = 0 \*/

#endif

#if (USBMCUTYPE == USBMCU\_MB9B610)

FM3\_USBETHERNETCLK->UCCR = 0x00;

FM3\_USBETHERNETCLK->USBEN1 = 0x00; /\* USBEN = 0 \*/

#endif

…

…

…

#if (USBMCUTYPE == USBMCU\_MB9B610)

FM3\_USBETHERNETCLK->UPCR1 = 0x01; /\* PLL Enable \*/

while(!FM3\_USBETHERNETCLK->UP\_STR);

bFM3\_USBETHERNETCLK\_UCCR\_UCSEL0 = 1;

bFM3\_USBETHERNETCLK\_UCCR\_UCSEL1 = 0;

bFM3\_USBETHERNETCLK\_UCCR\_UCEN1 = 1;

/\* wait over 5clock \*/

dummy = FM3\_USBETHERNETCLK->UCCR;

dummy = FM3\_USBETHERNETCLK->UCCR;

dummy = FM3\_USBETHERNETCLK->UCCR;

dummy = FM3\_USBETHERNETCLK->UCCR;

dummy = FM3\_USBETHERNETCLK->UCCR;

FM3\_USBETHERNETCLK->USBEN1 = 0x01; /\* USBEN = 1 \*/

#endif

}

#endif

…

…

…

#if IS\_USBMCU\_FM3

#ifdef USE\_USB0

void USB0\_Handler (void)

#else

void USB1\_Handler (void)

#endif

{

if (IS\_SET(HCNT, \_HCNT\_HOST))

{

#if (USE\_USB\_HOST == 1)

UsbHost\_ISR();

#endif

}

else

{

#if (USE\_USB\_FUNCTION == 1)

UsbFunction\_StatusControlIsr();

#endif

}

}

#endif

**(4)** spansion\usb\UsbInit.h を変更する

#ifndef \_\_USBINIT\_H\_\_

#define \_\_USBINIT\_H\_\_

#ifndef INCLUDE\_FROM\_CHIP\_H

#include "mcu.h"

#endif

#include "../base\_type.h"

#include "../usbfunction/UsbFunction.h"

…

…

…

#if defined(\_MB9BF506N\_H\_) || defined(\_MB9BF505N\_H\_) || defined(\_MB9BF504R\_H\_) || defined(\_MB9BF506R\_H\_) || defined(\_MB9BF505R\_H\_) || defined(\_MB9BF504R\_H\_)

#define USBMCUTYPE USBMCU\_MB9B500

#endif

#ifndef \_MB9B610T\_H\_

#define \_MB9B610T\_H\_

#endif

#if defined(\_MB9B610T\_H\_)

#ifndef USE\_USB1

#define USE\_USB1

#endif

#define USBMCUTYPE USBMCU\_MB9B610

#define \_\_IAR\_SYSTEMS\_ICC\_\_

#endif

…

…

…

#if IS\_USBMCU\_FM3

#define UDCC FM3\_USB1->UDCC

#define EP0C FM3\_USB1->EP0C

#define EP1C FM3\_USB1->EP1C

#define EP2C FM3\_USB1->EP2C

#define EP3C FM3\_USB1->EP3C

#define EP4C FM3\_USB1->EP4C

#define EP5C FM3\_USB1->EP5C

#define TMSP FM3\_USB1->TMSP

#define UDCS FM3\_USB1->UDCS

#define UDCIE FM3\_USB1->UDCIE

#define EP0IS FM3\_USB1->EP0IS

#define EP0OS FM3\_USB1->EP0OS

#define EP1S FM3\_USB1->EP1S

#define EP2S FM3\_USB1->EP2S

#define EP3S FM3\_USB1->EP3S

#define EP4S FM3\_USB1->EP4S

#define EP5S FM3\_USB1->EP5S

#define EP0DT FM3\_USB1->EP0DT

#define EP0DTL FM3\_USB1->EP0DTL

#define EP0DTH FM3\_USB1->EP0DTH

#define EP1DT FM3\_USB1->EP1DT

#define EP1DTL FM3\_USB1->EP1DTL

#define EP1DTH FM3\_USB1->EP1DTH

#define EP2DT FM3\_USB1->EP2DT

#define EP2DTL FM3\_USB1->EP2DTL

#define EP2DTH FM3\_USB1->EP2DTH

#define EP3DT FM3\_USB1->EP3DT

#define EP3DTL FM3\_USB1->EP3DTL

#define EP3DTH FM3\_USB1->EP3DTH

#define EP4DT FM3\_USB1->EP4DT

#define EP4DTL FM3\_USB1->EP4DTL

#define EP4DTH FM3\_USB1->EP4DTH

#define EP5DT FM3\_USB1->EP5DT

#define EP5DTL FM3\_USB1->EP5DTL

#define EP5DTH FM3\_USB1->EP5DTH

#define HCNT FM3\_USB1->HCNT

#define HIRQ FM3\_USB1->HIRQ

#define HERR FM3\_USB1->HERR

#define HSTATE FM3\_USB1->HSTATE

#define HFCOMP FM3\_USB1->HFCOMP

#define HRTIMER0 FM3\_USB1->HRTIMER0

#define HRTIMER1 FM3\_USB1->HRTIMER1

#define HRTIMER2 FM3\_USB1->HRTIMER2

#define HADR FM3\_USB1->HADR

#define HEOF (FM3\_USB1->HEOF)

#define HFRAME FM3\_USB1->HFRAME

#define HTOKEN FM3\_USB1->HTOKEN

#endif

**(5)** spansion\usb\UsbSofTimeout.hを変更する

#ifndef \_\_USBSOFTIMEOUT\_H\_\_

#define \_\_USBSOFTIMEOUT\_H\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "../base\_type.h"

**(6)** spansion\usbfunction\UsbDma.hを変更する

#ifndef \_\_USBDMA\_H\_\_

#define \_\_USBDMA\_H\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "../base\_type.h"

//#include "mcu.h"

#include "UsbFunction.h"

**(7)** spansion\usbfunction\UsbFunction.hを変更する

#ifndef \_\_USBFUNCTION\_H\_\_

#define \_\_USBFUNCTION\_H\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//#include "Usb.h"

#include "../usb/Usb\_fm3.h"

#if (USE\_USB\_FUNCTION == 1)

**(8)** spansion\usbhost\ClassDriverTable.hを変更する

#ifndef \_\_CLASSDRIVERTABLE\_H\_\_

#define \_\_CLASSDRIVERTABLE\_H\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//#include "Usb.h"

#include "../usb/Usb\_fm3.h"

#if (USE\_USB\_HOST == 1)

//#include "UsbMassStorage.h"

//#include "HidMouse.h"

//#include "HidKeyboard.h"

/\* OTHER USB CLASS DRIVER INCLUDES CAN BE ADDED HERE \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Global pre-processor symbols/macros ('#define') \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* DEFINEPARSER(DRIVERTYPES) START \*/

#define USBCLASSDRIVER\_MASSSTORAGE 1

#define USBCLASSDRIVER\_MOUSE 2

#define USBCLASSDRIVER\_JOYSTICK 3

#define USBCLASSDRIVER\_KEYBOARD 4

/\* DEFINEPARSER(DRIVERTYPES) STOP \*/

extern boolean\_t MassStorageInitHandler(uint8\_t\* pu8Configuration, uint32\_t u32Length);

extern boolean\_t MassStorageDeinitHandler();

extern boolean\_t HidMouseInitHandler(uint8\_t\* pu8Configuration, uint32\_t u32Length);

extern boolean\_t HidMouseDeinitHandler();

extern boolean\_t HidKeyboardInitHandler(uint8\_t\* pu8Configuration, uint32\_t u32Length);

extern boolean\_t HidKeyboardDeinitHandler();

/\* OTHER USB CLASS DRIVER DEFINES CAN BE ADDED HERE \*/

**(9)** spansion\usbhost\UsbClassSupervisor.cを変更する

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define \_\_USBCLASSSUPERVISOR\_C\_\_

#include "UsbClassSupervisor.h"

#if (USE\_USB\_HOST == 1)

//#include "HidMouse.h"

**(10)** spansion\usbhost\UsbClassSupervisor.hを変更する

#ifndef \_\_USBCLASSSUPERVISOR\_H\_\_

#define \_\_USBCLASSSUPERVISOR\_H\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//#include "Usb.h"

#include "../usb/Usb\_fm3.h"

#if (USE\_USB\_HOST == 1)

#include "../usb/UsbSofTimeout.h"

#include "../base\_type.h"

#include "../dbg/dbg.h"

**(11)** spansion\usbhost\UsbHost.cを変更する

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function implementation - global ('extern') and local ('static') \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void UsbHost\_Init()

{

volatile uint32\_t u32WaitTime = 500000;

bSofToken = FALSE;

u32ConTimeout = 0;

u32BusResetTimeout = 0;

u8CurrentEndpointPosition = 2;

u8EndpointListEnd = 0;

bSchedulerLock = FALSE;

bBusResetSend = FALSE;

SOFHandler = 0;

SetupCompletionHandler = 0;

u8DeviceStatus = USBHOST\_DEVICE\_IDLE;

u8DeviceAddress = 0;

u8EnumerationCounter = 0;

CLEAR\_MASK(HCNT, \_HCNT\_HOST); // Disable Host (HWM rev16 - Chapter 30.4.1)

HOSTVBUS\_INIT;

//HOSTOTGPULLDOWN\_INIT;

//HOSTVBUS\_SET;

HOSTVBUS\_CLEAR;

//HOSTOTGPULLDOWN\_SET;

while(u32WaitTime--) \_\_wait\_nop();

DBGOUT(1,dbg("UsbHost: Starting init... "));

**(12)** spansion\usbhost\UsbHost.hを変更する

#ifndef \_\_USBHOST\_H\_\_

#define \_\_USBHOST\_H\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//#include "Usb.h"

#include "../usb/Usb\_fm3.h"

#if (USE\_USB\_HOST == 1)

//#include "mcu.h"

#include "../usb/UsbInit.h"

#include "../base\_type.h"

#include "../dbg/dbg.h"

//#include "UsbHost.h"

#include "../usb/UsbSofTimeout.h"

#include "../usb/UsbSpec.h"

#include "../usb/UsbRegisters.h"

#include "UsbHostHW.h"

**(13)** spansion\usbhost\UsbHostHW.hを変更する

#ifndef \_\_USBHOSTHW\_H\_\_

#define \_\_USBHOSTHW\_H\_\_

#define HOSTVBUS\_INIT FM3\_GPIO->DDRF |= (1 << 5)

#define HOSTVBUS\_SET FM3\_GPIO->PDORF |= (1 << 5)

#define HOSTVBUS\_CLEAR FM3\_GPIO->PDORF &= ~(1 << 5)

#define HOSTVBUS\_ENABLED 1

#define HOSTOTGPULLDOWN\_INIT FM3\_GPIO->DDR1 |= (1 << 0x02)

#define HOSTOTGPULLDOWN\_SET FM3\_GPIO->PDOR1 |= (1 << 0x02)

#define HOSTOTGPULLDOWN\_CLEAR FM3\_GPIO->PDOR1 &= ~(1 << 0x02)

#define HOSTOTGPULLDOWN\_ENABLED 1

**(14)** D:\Arduino\hardware\arduino\sam\system\libsam\chip.h (line69)を変更する

#if (SAM3XA\_SERIES)

#include "include/can.h"

#include "include/emac.h"

#include "include/trng.h"

#include "include/uotghs\_device.h"

#include "include/uotghs\_host.h"

#endif /\* (SAM3XA\_SERIES) \*/

/\* FM3 UART and USB \*/

#define INCLUDE\_FROM\_CHIP\_H

#include "include/uart.h"

#include "source/spansion/base\_type.h"

#include "source/spansion/usb/Usb\_fm3.h"

#include "source/spansion/usbhost/UsbClassSupervisor.h"

#undef INCLUDE\_FROM\_CHIP\_H

#endif /\* \_LIB\_SAM\_ \*/

**(15)** D:\Arduino\hardware\arduino\sam\system\libsam\build\_gcc\sam3.mk (line73)を変更する

**(16)** 下記のファイルを削除する：

#-------------------------------------------------------------------------------

# Files

#-------------------------------------------------------------------------------

vpath %.h $(PROJECT\_BASE\_PATH)/include $(CMSIS\_ATMEL\_PATH) $(CMSIS\_CHIP\_PATH)/include

vpath %.c $(PROJECT\_BASE\_PATH)/source $(CMSIS\_ARM\_PATH) $(CMSIS\_CHIP\_PATH)/source

VPATH+=$(PROJECT\_BASE\_PATH)/source

VPATH+=$(PROJECT\_BASE\_PATH)/source/spansion

VPATH+=$(PROJECT\_BASE\_PATH)/source/spansion/dbg

VPATH+=$(PROJECT\_BASE\_PATH)/source/spansion/usb

VPATH+=$(PROJECT\_BASE\_PATH)/source/spansion/usbfunction

VPATH+=$(PROJECT\_BASE\_PATH)/source/spansion/usbhost

VPATH+=$(CMSIS\_ARM\_PATH)

VPATH+=$(CMSIS\_CHIP\_PATH)/include

VPATH+=$(CMSIS\_CHIP\_PATH)/source/

VPATH+=$(CMSIS\_CHIP\_PATH)/source/gcc

INCLUDES = -I$(PROJECT\_BASE\_PATH)

INCLUDES += -I$(PROJECT\_BASE\_PATH)/include

INCLUDES += -I$(PROJECT\_BASE\_PATH)/source/spansion

INCLUDES += -I$(PROJECT\_BASE\_PATH)/source/spansion/dbg

INCLUDES += -I$(PROJECT\_BASE\_PATH)/source/spansion/usb

INCLUDES += -I$(PROJECT\_BASE\_PATH)/source/spansion/usbfunction

INCLUDES += -I$(PROJECT\_BASE\_PATH)/source/spansion/usbhost

INCLUDES += -I$(CMSIS\_ARM\_PATH)

INCLUDES += -I$(CMSIS\_ATMEL\_PATH)

INCLUDES += -I$(CMSIS\_CHIP\_PATH)/include

…

…

…

#-------------------------------------------------------------------------------

# C source files and objects

#-------------------------------------------------------------------------------

C\_SRC=$(wildcard $(PROJECT\_BASE\_PATH)/source/\*.c)

C\_SRC+=$(wildcard $(PROJECT\_BASE\_PATH)/source/spansion/\*.c)

C\_SRC+=$(wildcard $(PROJECT\_BASE\_PATH)/source/spansion/dbg/\*.c)

C\_SRC+=$(wildcard $(PROJECT\_BASE\_PATH)/source/spansion/usb/\*.c)

C\_SRC+=$(wildcard $(PROJECT\_BASE\_PATH)/source/spansion/usbfunction/\*.c)

C\_SRC+=$(wildcard $(PROJECT\_BASE\_PATH)/source/spansion/usbhost/\*.c)

C\_SRC+=$(wildcard $(CMSIS\_CHIP\_PATH)/source/\*.c)

C\_SRC+=$(wildcard $(CMSIS\_CHIP\_PATH)/source/gcc/\*.c)

C\_OBJ\_TEMP=$(patsubst %.c, %.o, $(notdir $(C\_SRC)))

D:\Arduino\hardware\arduino\sam\system\CMSIS\Device\ATMEL\sam3xa\include\system\_mb9bf61x.h

下記のファイルを

sk-fm3-48pmc-usbstick\_usb\_host\_mouse\_keyboard\_massstorage-v10\common\ system\_mb9af31x.h

以下のファイルにコピーする。

D:\Arduino\hardware\arduino\sam\system\CMSIS\Device\ATMEL\sam3xa\include

D:\Arduino\hardware\arduino\sam\system\CMSIS\Device\ATMEL\sam3xa\include\mb9b610t.h (line135)を変更する

USB0F\_USB0H\_IRQn = 35, /\* USB Function ch.0 / USB Host ch.0 \*/

USB1F\_IRQn = 36, /\* USB Function ch.1 \*/

USB1F\_USB1H\_IRQn = 37, /\* USB Function ch.1 / USB Host ch.1 \*/

DMAC0\_IRQn = 38, /\* DMAC ch.0 \*/

DMAC1\_IRQn = 39, /\* DMAC ch.1 \*/

DMAC2\_IRQn = 40, /\* DMAC ch.2 \*/

DMAC3\_IRQn = 41, /\* DMAC ch.3 \*/

DMAC4\_IRQn = 42, /\* DMAC ch.4 \*/

DMAC5\_IRQn = 43, /\* DMAC ch.5 \*/

DMAC6\_IRQn = 44, /\* DMAC ch.6 \*/

DMAC7\_IRQn = 45, /\* DMAC ch.7 \*/

BTIM8\_15\_IRQn = 46 /\* Base Timer ch.8 to ch.15 \*/

/\* Reserved = 47 \*/

} IRQn\_Type;

#include <core\_cm3.h>

//#include "system\_mb9bf61x.h"

#include <stdint.h>

#define SUCCESS 0

#define ERROR -1

#ifndef NULL

#define NULL 0

#endif

D:\Arduino\hardware\arduino\sam\system\CMSIS\Device\ATMEL\sam3xa\include\mcu.hを変更する

**(17)** 下記のファイルを削除する：

#ifndef \_MB9B610T\_H\_

#include "mb9b610t.h"

#endif

#ifndef \_SYSTEM\_MB9AF31X\_H\_

#include "system\_mb9af31x.h"

#endif

D:\Arduino\hardware\arduino\sam\system\CMSIS\Device\ATMEL\sam3xa\source\system\_mb9bf61x.c

下記のファイルを

sk-fm3-48pmc-usbstick\_usb\_host\_mouse\_keyboard\_massstorage-v10\common\ system\_mb9af31x.c

以下のファイルにコピーする

D:\Arduino\hardware\arduino\sam\system\CMSIS\Device\ATMEL\sam3xa\source

D:\Arduino\hardware\arduino\sam\system\CMSIS\Device\ATMEL\sam3xa\source\system\_mb9af31x.を変更する

(line187)

#if (CR\_TRIM\_SETUP)

/\* CR Trimming Data \*/

if( 0x000003FF != (FM3\_FLASH\_IF->CRTRMM & 0x000003FF) )

{

/\* UnLock (MCR\_FTRM) \*/

FM3\_CRTRIM->MCR\_RLR = 0x1ACCE554;

/\* Set MCR\_FTRM \*/

FM3\_CRTRIM->MCR\_FTRM = FM3\_FLASH\_IF->CRTRMM;

/\* Lock (MCR\_FTRM) \*/

FM3\_CRTRIM->MCR\_RLR = 0x00000000;

}

#endif // (CR\_TRIM\_SETUP)

/\* Update the System Core Clock \*/

SystemCoreClockUpdate();

D:\Arduino\hardware\arduino\sam\system\CMSIS\Device\ATMEL\sam3xa\source\system\_sam3xa.cを変更する

#include "sam3xa.h"

/\* @cond 0 \*/

/\*\*INDENT-OFF\*\*/

#ifdef \_\_cplusplus

extern "C" {

#endif

/\*\*INDENT-ON\*\*/

/\* @endcond \*/

#if 0

/\* Clock settings (84MHz) \*/

#define SYS\_BOARD\_OSCOUNT (CKGR\_MOR\_MOSCXTST(0x8))

…

…

…

/\*\*

\* Initialize flash.

\*/

void system\_init\_flash(uint32\_t dw\_clk)

{

/\* Set FWS for embedded Flash access according to operating frequency \*/

if (dw\_clk < CHIP\_FREQ\_FWS\_0) {

EFC0->EEFC\_FMR = EEFC\_FMR\_FWS(0);

EFC1->EEFC\_FMR = EEFC\_FMR\_FWS(0);

} else if (dw\_clk < CHIP\_FREQ\_FWS\_1) {

EFC0->EEFC\_FMR = EEFC\_FMR\_FWS(1);

EFC1->EEFC\_FMR = EEFC\_FMR\_FWS(1);

} else if (dw\_clk < CHIP\_FREQ\_FWS\_2) {

EFC0->EEFC\_FMR = EEFC\_FMR\_FWS(2);

EFC1->EEFC\_FMR = EEFC\_FMR\_FWS(2);

} else if (dw\_clk < CHIP\_FREQ\_FWS\_3) {

EFC0->EEFC\_FMR = EEFC\_FMR\_FWS(3);

EFC1->EEFC\_FMR = EEFC\_FMR\_FWS(3);

} else {

EFC0->EEFC\_FMR = EEFC\_FMR\_FWS(4);

EFC1->EEFC\_FMR = EEFC\_FMR\_FWS(4);

}

}

#endif

**(18)** D:\Arduino\hardware\arduino\sam\system\CMSIS\Device\ATMEL\sam3xa\source\gcc\startup\_sam3xa.cを変更する

(line30)

/\* Initialize segments \*/

extern uint32\_t \_sfixed;

extern uint32\_t \_efixed;

extern uint32\_t \_etext;

extern uint32\_t \_srelocate;

extern uint32\_t \_erelocate;

extern uint32\_t \_szero;

extern uint32\_t \_ezero;

extern uint32\_t \_sstack;

extern uint32\_t \_estack;

extern void MFS0RX\_IRQHandler(void);

extern void USB1\_Handler(void);

…

…

…

#ifdef \_SAM3XA\_TC2\_INSTANCE\_

(void\*) TC6\_Handler, /\* 33 Timer Counter 6 \*/

(void\*) TC7\_Handler, /\* 34 Timer Counter 7 \*/

(void\*) TC8\_Handler, /\* 35 Timer Counter 8 \*/

#else

(void\*) (0UL), /\* 33 Reserved \*/

(void\*) (0UL), /\* 34 Reserved \*/

(void\*) (0UL), /\* 35 Reserved \*/

#endif /\* \_SAM3XA\_TC2\_INSTANCE\_ \*/

(void\*) PWM\_Handler, /\* 36 PWM \*/

#if 0

(void\*) ADC\_Handler, /\* 37 ADC controller \*/

#endif

(void\*) USB1\_Handler, /\* 37 FM3 USB1 HOST \*/

(void\*) DACC\_Handler, /\* 38 DAC controller \*/

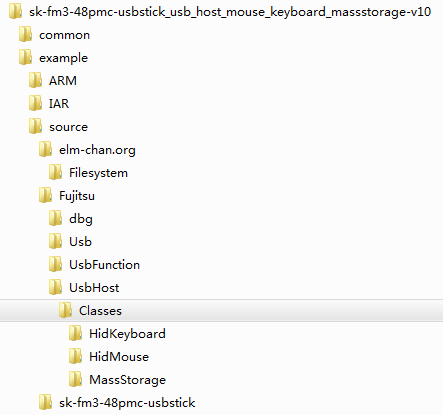
(void\*) DMAC\_Handler, /\* 39 DMA Controller \*/

(void\*) UOTGHS\_Handler, /\* 40 USB OTG High Speed \*/

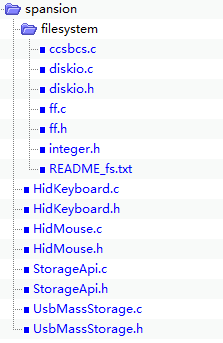
(void\*) TRNG\_Handler, /\* 41 True Random Number Generator \*/

# 5．USB Host Applicationの対応

USB Host ApplicationのソースはSK-FM3-48PMC-USBSTICK光ディスクの以下のディレクトリより取得されたものである：CD\Examples\sk-fm3-48pmc-usbstick\_usb\_host\_mouse\_keyboard\_massstorage-v10



D:\Arduino\libraries\USBHost\srcパスの配下にspansionフォルダを新規作成する。ディレクトリ構成を下図に示す。ファイルは上記のフォルダより取得されたものである。



**(1)** spansion\HidKeyboard.cを変更する

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define \_\_HIDKEYBOARD\_C\_\_

//#include "Usb.h"

#include "Arduino.h"

#if (USE\_USB\_HOST == 1)

#include "HidKeyboard.h"

…

…

…

void HidKeyboard\_DataReceivedHandler()

{

uint8\_t i = 0;

boolean\_t bExecuteEvent = FALSE;

for(i=0;i<8;i++)

{

if ((bExecuteEvent == FALSE) && ((EndpointIN->pu8Buffer[i] != 0) || ((EndpointIN->pu8Buffer[i] == 0) && (pu8OldData[i] != EndpointIN->pu8Buffer[i]))))

{

bExecuteEvent = TRUE;

}

pu8OldData[i] = EndpointIN->pu8Buffer[i];

}

stcKeyboardData.u8ModifierKeys = EndpointIN->pu8Buffer[0];

stcKeyboardData.u8Reserved = EndpointIN->pu8Buffer[1];

stcKeyboardData.u8KeyCode1 = EndpointIN->pu8Buffer[2];

stcKeyboardData.u8KeyCode2 = EndpointIN->pu8Buffer[3];

stcKeyboardData.u8KeyCode3 = EndpointIN->pu8Buffer[4];

stcKeyboardData.u8KeyCode4 = EndpointIN->pu8Buffer[5];

stcKeyboardData.u8KeyCode5 = EndpointIN->pu8Buffer[6];

stcKeyboardData.u8KeyCode6 = EndpointIN->pu8Buffer[7];

bKeyboardChanged = bExecuteEvent;

HidDataCopy(8,EndpointIN->pu8Buffer);

if ((bExecuteEvent == TRUE) && (KeyEventHandler != 0))

{

KeyEventHandler(&stcKeyboardData);

}

EndpointIN->CompletionHandler = HidKeyboard\_DataReceivedHandler;

}

**(2)** spansion\HidKeyboard.cを変更する

#ifndef \_\_HIDKEYBOARD\_H\_\_

#define \_\_HIDKEYBOARD\_H\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//#include "mcu.h"

//#include "base\_type.h"

//#include "UsbHost.h"

#include "Arduino.h"

#if (USE\_USB\_HOST == 1)

#include "HidKeyboard.h"

…

…

…

void HidKeyboard\_RemoveEvent();

void UMH\_StallExecutionUserCallback(unsigned long time);

void Idle(void);

extern void HidDataCopy(uint32\_t len, uint8\_t \*buf);

#endif /\* (USE\_USB\_HOST == 1) \*/

#endif /\* \_\_HidKeyboard\_H\_\_\*/

**(3)** spansion\HidMouse.cを変更する

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//#include "Usb.h"

#include "Arduino.h"

#if (USE\_USB\_HOST == 1)

#include "HidMouse.h"

…

…

…

void HidMouse\_DataReceivedHandler()

{

MouseEventType\_t tEvent = 0;

bMouseMoved = TRUE;

if (EndpointIN->pu8Buffer[0] != stcMouseData.tButtons)

{

tEvent |= MOUSEEVENT\_BUTTON\_CHANGED;

}

…

…

…

if (stcMouseData.i8Scrolling < 0)

{

stcMouseData.u32ScrollPosition--;

}

HidDataCopy(4,EndpointIN->pu8Buffer);

if ((tEvent) && (MouseEventHandler))

{

MouseEventHandler(tEvent,stcMouseData);

}

**(4)** spansion\HidMouse.hを変更する

#ifndef \_\_HIDMOUSE\_H\_\_

#define \_\_HIDMOUSE\_H\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//#include "mcu.h"

//#include "base\_type.h"

//#include "UsbHost.h"

#include "Arduino.h"

#if (USE\_USB\_HOST == 1)

#include "HidMouse.h"

…

…

…

void UMH\_StallExecutionUserCallback(unsigned long time);

void Idle(void);

extern void HidDataCopy(uint32\_t len, uint8\_t \*buf);

#endif /\* (USE\_USB\_HOST == 1) \*/

#endif /\* \_\_HIDMOUSE\_H\_\_\*/

**(5)** spansion\StorageApi.cを変更する

//#include "Usb.h"

#include "Arduino.h"

#if (USE\_USB\_HOST == 1)

#include <stdlib.h>

#include "UsbMassStorage.h"

**(6)** spansion\StorageApi.hを変更する

#ifndef \_STORAGE\_API\_H\_

#define \_STORAGE\_API\_H\_

//#include "Usb.h"

#include "Arduino.h"

#if (USE\_USB\_HOST == 1)

#include "filesystem/diskio.h"

//defines

//UFI Command Descriptors (SCSI)

**(7)** spansion\UsbMassStorage.cを変更する

Arduino固有関数にsetupと同名する関数がすでに存在しているため、setupをsetupmscに変更する。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//#include "Usb.h"

#include "Arduino.h"

#if (USE\_USB\_HOST == 1)

#include "UsbMassStorage.h"

…

…

…

volatile static uint8\_t u8Status = 0;

volatile static boolean\_t bSetupComplete = FALSE;

StorageInfo\_t stcGlobalStorageInfo;

static UsbRequest\_t setupmsc;

…

…

…

int16\_t UsbMassStorage\_BulkOnlyMSReset(void)

{

ZERO\_STRUCT(setupmsc);

setupmsc.bmRequestType=0x21;

setupmsc.bRequest=0xFF;

setupmsc.wValue=0;

setupmsc.wIndex=0;

setupmsc.wLength=0;

UsbHost\_SetupRequest(&setupmsc,MassStorageSetupComplete);

MassStorageWaitForSetupComplete();

return USBH\_STATUS\_SUCCESS;

}

…

…

…

int16\_t UsbMassStorage\_ClearInEndpoint(void)

{

ZERO\_STRUCT(setupmsc);

setupmsc.bmRequestType=0x02;

setupmsc.bRequest=0x01;

setupmsc.wValue=0x00;

setupmsc.wIndex = USB\_IN\_DIRECTION | u8EndpointIN;

setupmsc.wLength=0x00;

//UMH\_AbortTransfer(umhInHandle); TBD

UsbHost\_SetupRequest(&setupmsc,MassStorageSetupComplete);

MassStorageWaitForSetupComplete();

//UMH\_ResetDataToggleBit(umhInHandle); TBD

return USBH\_STATUS\_SUCCESS;

}

int16\_t UsbMassStorage\_ClearOutEndpoint(void)

{

ZERO\_STRUCT(setupmsc);

setupmsc.bmRequestType=0x02;

setupmsc.bRequest=0x01;

setupmsc.wValue=0x00;

setupmsc.wIndex=USB\_OUT\_DIRECTION | u8EndpointOUT;

setupmsc.wLength=0x00;

//UMH\_AbortTransfer(umhOutHandle); TBD

UsbHost\_SetupRequest(&setupmsc,MassStorageSetupComplete);

MassStorageWaitForSetupComplete();

//UMH\_ResetDataToggleBit(umhOutHandle); TBD

return USBH\_STATUS\_SUCCESS;

}

**(8)** spansion\ UsbMassStorage.hを変更する

#ifndef \_\_USBMASSSTORAGE\_H\_\_

#define \_\_USBMASSSTORAGE\_H\_\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Include files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//#include "mcu.h"

//#include "base\_type.h"

//#include "UsbHost.h"

#include "Arduino.h"

#if (USE\_USB\_HOST == 1)

//#include "usbspec.h"

#include "UsbMassStorage.h"

//#include "StorageApi.h"

**(9)** spansion\filesystem\diskio.cを変更する

#include "diskio.h"

#include "../UsbMassStorage.h"

#include "../StorageApi.h"

//#include "usb.h"

#include "Arduino.h"

#if (USE\_USB\_HOST == 1)

**(10)** spansion\filesystem\ff.cを変更する

#include "ff.h" /\* FatFs configurations and declarations \*/

#include "diskio.h" /\* Declarations of low level disk I/O functions \*/

//#include "usb.h"

#include "Arduino.h"

#if (USE\_USB\_HOST == 1)

**(11)** spansion\filesystem\ff.hを変更する

#define \_FS\_RPATH 1

/\* When \_FS\_RPATH is set to 1, relative path feature is enabled and f\_chdir,

/ f\_chdrive function are available.

/ Note that output of the f\_readdir fnction is affected by this option. \*/

**(12)** spansion\filesystem\integer.hを変更する

/\*-------------------------------------------\*/

/\* Integer type definitions for FatFs module \*/

/\*-------------------------------------------\*/

//#include "base\_type.h"

#include "Arduino.h"

#ifndef \_INTEGER

**(13)** D:\Arduino\libraries\USBHost\src\hidboot.cppを変更する

#include "hidboot.h"

#define HIDDATABUF\_SIZE 16

uint32\_t HidDataNum = 0;

uint8\_t HidDataBuf[HIDDATABUF\_SIZE];

extern "C" {

void HidDataCopy(uint32\_t len, uint8\_t \*buf) {

if (!HidDataNum)

{

if (len > HIDDATABUF\_SIZE)

len = HIDDATABUF\_SIZE;

HidDataNum = len;

while(--len){

HidDataBuf[len] = buf[len];

}

HidDataBuf[0] = buf[0];

}

}

}

/\*\*

\* \brief Parse HID mouse report.

\*

\* \param hid HID device pointer.

\* \param is\_rpt\_id True if this is a report ID.

\* \param len Buffer length.

\* \param buf Buffer containing report data.

\*/

void MouseReportParser::Parse(HID \*hid, bool is\_rpt\_id, uint32\_t len, uint8\_t \*buf)

{

MOUSEINFO \*pmi = (MOUSEINFO\*)buf;

if (prevState.mouseInfo.bmLeftButton == 0 && pmi->bmLeftButton == 1)

OnLeftButtonDown(pmi);

if (prevState.mouseInfo.bmLeftButton == 1 && pmi->bmLeftButton == 0)

OnLeftButtonUp(pmi);

if (prevState.mouseInfo.bmRightButton == 0 && pmi->bmRightButton == 1)

OnRightButtonDown(pmi);

if (prevState.mouseInfo.bmRightButton == 1 && pmi->bmRightButton == 0)

OnRightButtonUp(pmi);

if (prevState.mouseInfo.bmMiddleButton == 0 && pmi->bmMiddleButton == 1)

OnMiddleButtonDown(pmi);

if (prevState.mouseInfo.bmMiddleButton == 1 && pmi->bmMiddleButton == 0)

OnMiddleButtonUp(pmi);

if (prevState.mouseInfo.dX != pmi->dX || prevState.mouseInfo.dY != pmi->dY)

OnMouseMove(pmi);

if (pmi->dZ != 0)

OnMouseScroll(pmi);

for (uint32\_t i = 0; i < 3; ++i)

prevState.bInfo[i] = buf[i];

};

**(14)** D:\Arduino\libraries\USBHost\src\hidboot.hを変更する

/\*\*

\* \brief MOUSEINFO definition.

\*/

struct MOUSEINFO

{

struct

{

uint8\_t bmLeftButton : 1;

uint8\_t bmRightButton : 1;

uint8\_t bmMiddleButton : 1;

uint8\_t bmDummy : 1;

};

int8\_t dX;

int8\_t dY;

int8\_t dZ;

};

/\*\*

\* \class MouseReportParser definition.

\*/

class MouseReportParser : public HIDReportParser

{

union

{

MOUSEINFO mouseInfo;

uint8\_t bInfo[3];

} prevState;

public:

virtual void Parse(HID \*hid, bool is\_rpt\_id, uint32\_t len, uint8\_t \*buf);

protected:

virtual void OnMouseScroll (MOUSEINFO \*mi) {};

virtual void OnMouseMove (MOUSEINFO \*mi) {};

virtual void OnLeftButtonUp (MOUSEINFO \*mi) {};

virtual void OnLeftButtonDown (MOUSEINFO \*mi) {};

virtual void OnRightButtonUp (MOUSEINFO \*mi) {};

virtual void OnRightButtonDown (MOUSEINFO \*mi) {};

virtual void OnMiddleButtonUp (MOUSEINFO \*mi) {};

virtual void OnMiddleButtonDown (MOUSEINFO \*mi) {};

};

…

…

…

extern uint32\_t HidDataNum;

extern uint8\_t HidDataBuf[];

/\*\*

\* \brief Poll USB device activity.

\*

\* \note Poll call is periodically made from USBHost.task().

\*

\* \return 0 on success, error code otherwise.

\*/

template <const uint8\_t BOOT\_PROTOCOL>

uint32\_t HIDBoot<BOOT\_PROTOCOL>::Poll()

{

uint32\_t rcode = 0;

#if 0

if (!bPollEnable)

return 0;

if (qNextPollTime <= millis())

{

qNextPollTime = millis() + 10;

const uint32\_t const\_buff\_len = 16;

uint8\_t buf[const\_buff\_len];

uint32\_t read = epInfo[epInterruptInIndex].maxPktSize;

rcode = pUsb->inTransfer(bAddress, epInfo[epInterruptInIndex].deviceEpNum, &read, buf);

if (rcode)

{

return rcode;

}

if (pRptParser)

pRptParser->Parse((HID\*)this, 0, (uint32\_t)read, buf);

}

#endif

if (HidDataNum)

{

if (pRptParser)

pRptParser->Parse((HID\*)this, 0, HidDataNum, HidDataBuf);

HidDataNum = 0;

}

return rcode;

}

#endif /\* HIDBOOT\_H\_INCLUDED \*/

**(15)** D:\Arduino\libraries\USBHost\src\MouseController.cppを変更する

#include <MouseController.h>

extern "C" {

void \_\_mouseControllerEmptyCallback() { }

}

void mouseScrolled() \_\_attribute\_\_ ((weak, alias("\_\_mouseControllerEmptyCallback")));

void mouseClicked() \_\_attribute\_\_ ((weak, alias("\_\_mouseControllerEmptyCallback")));

void mouseDragged() \_\_attribute\_\_ ((weak, alias("\_\_mouseControllerEmptyCallback")));

void mouseMoved() \_\_attribute\_\_ ((weak, alias("\_\_mouseControllerEmptyCallback")));

void mousePressed() \_\_attribute\_\_ ((weak, alias("\_\_mouseControllerEmptyCallback")));

void mouseReleased() \_\_attribute\_\_ ((weak, alias("\_\_mouseControllerEmptyCallback")));

int MouseController::getXChange() {

int r = dx;

dx = 0;

return r;

}

int MouseController::getYChange() {

int r = dy;

dy = 0;

return r;

}

int MouseController::getWheelChange() {

int r = dz;

dz = 0;

return r;

}

void MouseController::OnMouseScroll(MOUSEINFO \*mi) {

dz += mi->dZ;

mouseScrolled();

}

**(16)** D:\Arduino\libraries\USBHost\src\MouseController.hを変更する

class MouseController : public MouseReportParser

{

public:

MouseController(USBHost &usb) : hostMouse(&usb), dx(0), dy(0), buttons(0) {

hostMouse.SetReportParser(0, this);

};

bool getButton(MouseButton button) { return (buttons & button) == button; };

int getXChange();

int getYChange();

int getWheelChange();

protected:

virtual void OnMouseScroll(MOUSEINFO \*mi);

virtual void OnMouseMove(MOUSEINFO \*mi);

virtual void OnLeftButtonUp(MOUSEINFO \*mi);

virtual void OnLeftButtonDown(MOUSEINFO \*mi);

virtual void OnMiddleButtonUp(MOUSEINFO \*mi);

virtual void OnMiddleButtonDown(MOUSEINFO \*mi);

virtual void OnRightButtonUp(MOUSEINFO \*mi);

virtual void OnRightButtonDown(MOUSEINFO \*mi);

private:

HIDBoot<HID\_PROTOCOL\_MOUSE> hostMouse;

int dx, dy, dz;

int buttons;

};

**(17)** D:\Arduino\libraries\USBHost\src\Usb.cppを変更する

/\*\*

\* \brief USBHost class constructor.

\*/

USBHost::USBHost() : bmHubPre(0)

{

// Set up state machine

usb\_task\_state = USB\_DETACHED\_SUBSTATE\_INITIALIZE;

// Init host stack

init();

// Initialize fm3 usb host

UsbInit();

UsbHost\_Init();

}

…

…

…

/\*\*

\* \brief USB main task, responsible for enumeration and clean up stage.

\*

\* \note Must be periodically called from loop().

\*/

void USBHost::Task(void)

{

// Poll connected devices

for (uint32\_t i = 0; i < USB\_NUMDEVICES; ++i)

if (devConfig[i])

devConfig[i]->Poll();

#if 0

uint32\_t rcode = 0;

volatile uint32\_t tmpdata = 0;

static uint32\_t delay = 0;

uint32\_t lowspeed = 0;

// Update USB task state on Vbus change

tmpdata = UHD\_GetVBUSState();

switch (tmpdata)

…

…

…

case USB\_STATE\_RUNNING:

break;

case USB\_STATE\_ERROR:

break;

}

#endif

}

**(18)** D:\Arduino\libraries\USBHost\src\UsbStorageFatFs.hを新規作成する

#ifndef USBSTORAGE\_FATFS\_H

#define USBSTORAGE\_FATFS\_H

#include "Arduino.h"

#include "Usb.h"

extern "C" {

#include "spansion\StorageApi.h"

#include "spansion\filesystem\ff.h"

}

class UsbStorageFatFs : public USBDeviceConfig

{

protected:

USBHost \*pUsb;

public:

UsbStorageFatFs(USBHost &usb) : pUsb(&usb) {

if (pUsb)

pUsb->RegisterDeviceClass(this);

};

// USBDeviceConfig implementation

virtual uint32\_t Init(uint32\_t parent, uint32\_t port, uint32\_t lowspeed) { return 0; };

virtual uint32\_t Release() { return 0; };

virtual uint32\_t Poll();

virtual uint32\_t GetAddress() { return 0; };

/\* FatFs module application interface \*/

boolean begin(); /\* Begin USB disk \*/

DWORD gettotal(); /\* Get total size (MB) of USB disk \*/

DWORD getfree(); /\* Get Free size (MB) of USB disk \*/

boolean fopen(const XCHAR\*, BYTE); /\* Open or create a file \*/

UINT fread(void\*, UINT); /\* Read data from a file \*/

UINT fwrite(const void\*, UINT); /\* Write data to a file \*/

boolean flseek(DWORD); /\* Move file pointer of a file object \*/

boolean ftruncate(); /\* Truncate file \*/

boolean fsync(); /\* Flush cached data of a writing file \*/

boolean fclose(); /\* Close an open file object \*/

FILINFO\* fstat(const XCHAR\*); /\* Get file or directory status \*/

boolean rename(const XCHAR\*, const XCHAR\*); /\* Rename/Move a file or directory \*/

boolean remove(const XCHAR\*); /\* Delete an existing file \*/

boolean opendir(const XCHAR\*); /\* Open an existing directory \*/

FILINFO\* readdir(); /\* Read a directory item \*/

boolean chdir(const XCHAR\*); /\* Change current directory \*/

boolean mkdir(const XCHAR\*); /\* Create a new directory \*/

boolean rmdir(const XCHAR\*); /\* Delete an existing directory \*/

boolean exists(const XCHAR\*); /\* Is file or directory existing \*/

#if \_USE\_STRFUNC

int fputc(int); /\* Put a character to the file \*/

int fputs(const char\*); /\* Put a string to the file \*/

int fprintf(const char\*, ...); /\* Put a formatted string to the file \*/

char\* fgets(char\*, int); /\* Get a string from the file \*/

#endif

private:

FATFS fatfs[1];

FATFS \*fs;

FIL fil;

FILINFO finfo;

DIR dir;

DWORD clust;

};

#endif

**(19)** D:\Arduino\libraries\USBHost\src\UsbStorageFatFs.cppを新規作成する

#include <UsbStorageFatFs.h>

boolean UsbStorageFatFs::begin() {

boolean ret = FALSE;

if ((uint8\_t)UsbClassSupervisor\_GetActiveDriver() == USBCLASSDRIVER\_MASSSTORAGE)

{

delay(200);

while((StorageInit() != 0) && (UsbHost\_GetDeviceStatus() != USBHOST\_DEVICE\_IDLE))

{

UsbHost\_DeInit();

delay(200);

UsbHost\_Init();

delay(200);

}

if (UsbHost\_GetDeviceStatus() != USBHOST\_DEVICE\_IDLE)

{

f\_mount(0,fatfs);

if (f\_opendir(&dir, "0:") == FR\_OK)

{

ret = TRUE;

}

}

}

return ret;

}

DWORD UsbStorageFatFs::gettotal() {

DWORD ret = 0;

if (f\_getfree("0:", &clust, &fs) == FR\_OK)

ret = ((fs->max\_clust - 2) \* fs->csize) / 2;

return ret/1024;

}

DWORD UsbStorageFatFs::getfree() {

DWORD ret = 0;

if (f\_getfree("0:", &clust, &fs) == FR\_OK)

ret = (clust \* fs->csize) / 2;

return ret/1024;

}

boolean UsbStorageFatFs::fopen(const XCHAR \*path, BYTE mode) {

boolean ret = FALSE;

if (f\_open(&fil, path, mode) == FR\_OK)

ret = TRUE;

return ret;

}

UINT UsbStorageFatFs::fread(void \*buff, UINT btr) {

UINT ret;

if (f\_read(&fil, buff, btr, &ret) != FR\_OK)

ret = 0;

return ret;

}

UINT UsbStorageFatFs::fwrite(const void \*buff, UINT btw) {

UINT ret;

if (f\_write(&fil, buff, btw, &ret) != FR\_OK)

ret = 0;

return ret;

}

boolean UsbStorageFatFs::flseek(DWORD ofs) {

boolean ret = FALSE;

if (f\_lseek(&fil, ofs) == FR\_OK)

ret = TRUE;

return ret;

}

boolean UsbStorageFatFs::ftruncate() {

boolean ret = FALSE;

if (f\_truncate(&fil) == FR\_OK)

ret = TRUE;

return ret;

}

boolean UsbStorageFatFs::fsync() {

boolean ret = FALSE;

if (f\_sync(&fil) == FR\_OK)

ret = TRUE;

return ret;

}

boolean UsbStorageFatFs::fclose() {

boolean ret = FALSE;

if (f\_close(&fil) == FR\_OK)

ret = TRUE;

return ret;

}

FILINFO\* UsbStorageFatFs::fstat(const XCHAR \*path) {

FILINFO\* ret = NULL;

if (f\_stat(path, &finfo) == FR\_OK)

ret = &finfo;

return ret;

}

boolean UsbStorageFatFs::rename(const XCHAR \*path\_old, const XCHAR \*path\_new) {

boolean ret = FALSE;

if (f\_rename(path\_old, path\_new) == FR\_OK)

ret = TRUE;

return ret;

}

boolean UsbStorageFatFs::remove(const XCHAR \*path) {

boolean ret = FALSE;

if (f\_unlink(path) == FR\_OK)

ret = TRUE;

return ret;

}

boolean UsbStorageFatFs::opendir(const XCHAR \*path) {

boolean ret = FALSE;

if (f\_opendir(&dir, path) == FR\_OK)

ret = TRUE;

return ret;

}

FILINFO\* UsbStorageFatFs::readdir() {

FILINFO\* ret = NULL;

if (f\_readdir(&dir, &finfo) == FR\_OK)

ret = &finfo;

return ret;

}

boolean UsbStorageFatFs::chdir(const XCHAR \*path) {

boolean ret = FALSE;

if (f\_chdir(path) == FR\_OK)

ret = TRUE;

return ret;

}

boolean UsbStorageFatFs::mkdir(const XCHAR \*path) {

boolean ret = FALSE;

if (f\_mkdir(path) == FR\_OK)

ret = TRUE;

return ret;

}

boolean UsbStorageFatFs::rmdir(const XCHAR \*path) {

boolean ret = FALSE;

if (f\_unlink(path) == FR\_OK)

ret = TRUE;

return ret;

}

boolean UsbStorageFatFs::exists(const XCHAR \*filepath) {

boolean ret = FALSE;

FILINFO\* pfinfo;

if (!opendir("0:"))

return ret;

while ((pfinfo = readdir()) && pfinfo->fname[0])

{

if (strcmp((char\*)pfinfo->fname, filepath) == 0)

ret = TRUE;

}

return ret;

}

#if \_USE\_STRFUNC

int UsbStorageFatFs::fputc(int chr) {

return f\_putc(chr, &fil);

}

int UsbStorageFatFs::fputs(const char\* str) {

return f\_puts(str, &fil);

}

int UsbStorageFatFs::fprintf(const char\* str, ...) {

//return f\_printf(&fil, str, ...);

return 0;

}

char\* UsbStorageFatFs::fgets(char\* buff, int len) {

return f\_gets(buff, len, &fil);

}

#endif

uint32\_t UsbStorageFatFs::Poll() {

return 0;

}

**(20)** D:\Arduino\libraries\USBHost\examples\MouseController\MouseController.inoを変更する

// Require mouse control library

#include <MouseController.h>

// Initialize USB Controller

USBHost usb;

// Attach mouse controller to USB

MouseController mouse(usb);

// variables for mouse button states

boolean leftButton = false;

boolean middleButton = false;

boolean rightButton = false;

// This function intercepts mouse scroll

void mouseScrolled() {

Serial.print("Scroll: ");

Serial.println(mouse.getWheelChange());

}

// This function intercepts mouse movements

void mouseMoved() {

Serial.print("Move: ");

Serial.print(mouse.getXChange());

Serial.print(", ");

Serial.println(mouse.getYChange());

}

// This function intercepts mouse movements while a button is pressed

void mouseDragged() {

Serial.print("DRAG: ");

Serial.print(mouse.getXChange());

Serial.print(", ");

Serial.println(mouse.getYChange());

}

// This function intercepts mouse button press

void mousePressed() {

Serial.print("Pressed: ");

if (mouse.getButton(LEFT\_BUTTON)){

Serial.print("L");

leftButton = true;

}

if (mouse.getButton(MIDDLE\_BUTTON)){

Serial.print("M");

middleButton = true;

}

if (mouse.getButton(RIGHT\_BUTTON)){

Serial.print("R");

rightButton = true;

}

Serial.println();

}

**(21)** D:\Arduino\libraries\USBHost\examples\UsbStorageFatFs\UsbStorageFatFs.inoを新規作成する

// Require USB Storage FatFs library

#include <UsbStorageFatFs.h>

// Initialize USB Controller

USBHost usb;

// Attach usb storage fatfs to USB

UsbStorageFatFs udisk(usb);

#define TEST\_DIR "TEST\_DIR"

#define UPPER\_DIR ".."

#define TEST\_FILE "test.txt"

#define TEST\_STR "This is a test string writed to file!"

void setup()

{

Serial.begin(9600);

Serial.println("Program started");

delay(200);

while(!udisk.begin());

Serial.println("USB Disk is ready!");

Serial.print("Total size(MB):");

Serial.println(udisk.gettotal());

Serial.print("Free size(MB):");

Serial.println(udisk.getfree());

if(udisk.exists(TEST\_DIR))

{

Serial.println("Directory exists!");

if(udisk.chdir(TEST\_DIR))

{

Serial.println("chdir OK!");

if(udisk.exists(TEST\_FILE))

{

Serial.println("File exists!");

if(udisk.remove(TEST\_FILE))

Serial.println("Remove file OK!");

}

}

if(udisk.chdir(UPPER\_DIR))

{

Serial.println("Back to upper directory OK!");

if(udisk.exists(TEST\_DIR))

{

Serial.println("Directory exists!");

if(udisk.rmdir(TEST\_DIR))

Serial.println("Remove directory OK!");

}

}

}

if(udisk.exists(TEST\_DIR) || udisk.mkdir(TEST\_DIR))

{

Serial.println("TEST\_DIR OK!");

if(udisk.chdir(TEST\_DIR))

{

Serial.println("chdir OK!");

if(udisk.fopen(TEST\_FILE,FA\_READ | FA\_WRITE | FA\_CREATE\_NEW))

{

Serial.println("fopen OK!");

if(udisk.fwrite(TEST\_STR, sizeof(TEST\_STR) - 1))

Serial.println("fwrite OK!");

if(udisk.fclose())

Serial.println("fclose OK!");

}

}

}

}

void loop()

{

// Process USB tasks

usb.Task();

}

void loop()

{

// Process USB tasks

usb.Task();

}

**(22)** D:\Arduino\libraries\USBHost\keywords.txtを変更する

#######################################

# Syntax Coloring Map For USBHost

#######################################

#######################################

# Datatypes (KEYWORD1)

#######################################

MouseController KEYWORD1

USBHost KEYWORD1

KeyboardController KEYWORD1

UsbStorageFatFs KEYWORD1

#######################################

# Methods and Functions (KEYWORD2)

#######################################

Task KEYWORD2

mouseMoved KEYWORD2

mouseDragged KEYWORD2

mousePressed KEYWORD2

mouseReleased KEYWORD2

getXChange KEYWORD2

getYChange KEYWORD2

getWheelChange KEYWORD2

getButton KEYWORD2

keyPressed KEYWORD2

keyReleased KEYWORD2

getModifiers KEYWORD2

getKey KEYWORD2

getOemKey KEYWORD2

begin KEYWORD2

gettotal KEYWORD2

getfree KEYWORD2

fopen KEYWORD2

fread KEYWORD2

fwrite KEYWORD2

flseek KEYWORD2

ftruncate KEYWORD2

fsync KEYWORD2

fclose KEYWORD2

fstat KEYWORD2

rename KEYWORD2

remove KEYWORD2

opendir KEYWORD2

readdir KEYWORD2

chdir KEYWORD2

mkdir KEYWORD2

rmdir KEYWORD2

exists KEYWORD2

fputc KEYWORD2

fputs KEYWORD2

#######################################

# Constants (LITERAL1)

#######################################

# 6．工程のビルドおよび検証

**(1)** libsamライブラリファイルをビルドする

ＤＯＳ窓（コマンドプロンプト）を開いて、以下の操作を実行する。

ディレクトリ名は各自がarduinoを展開したディレクトリ名へ変更する必要がある。

cd D:\Arduino\hardware\arduino\sam\system\libsam\build\_gcc

cs-make clean

cs-make -f Makefile ARM\_GCC\_TOOLCHAIN=D:\Arduino\hardware\tools\g++\_arm\_none\_eabi\bin MAKE=cs-make arduino\_due\_x

ビルドで生成されたオブジェクトファイル（ ～.o ）が置かれるディレクトリ

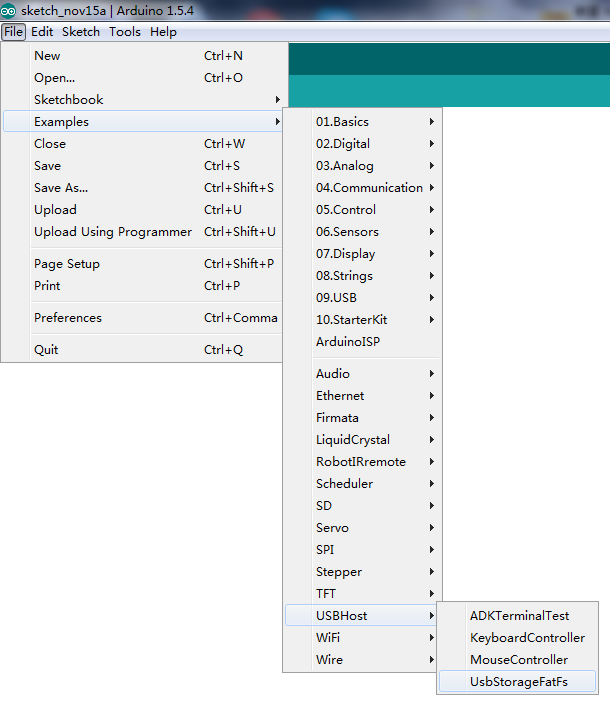
D:\Arduino\hardware\arduino\sam\system\libsam\build\_gcc\release\_sam3x8e

ビルドで生成されたライブラリ

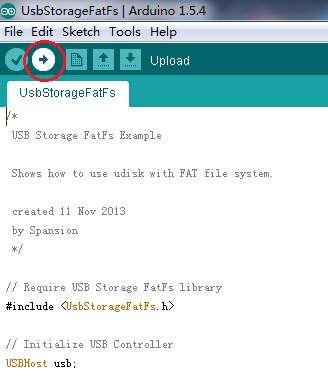
D:\Arduino\hardware\arduino\sam\variants\arduino\_due\_x\libsam\_sam3x8e\_gcc\_rel.a

**(2)** Arduino IDEにおいて UsbStorageFatFs例をビルドする

①まず、下図に示したようにUsbStorageFatFs例をビルドする

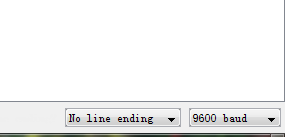
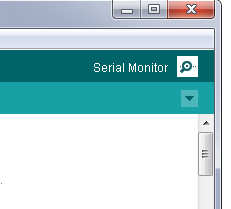


②Uploadボタンをクリックして、UsbStorageFatFs例をビルドする。



③MDK+J-LINK またはUSB DIRECT Programmerツールを利用して、ビルドで生成されたD:\Arduino\build\UsbStorageFatFs.cpp.hexファイルをIAR SYSTEMSの MB9BF618T-SK ボードにダウンロードする。

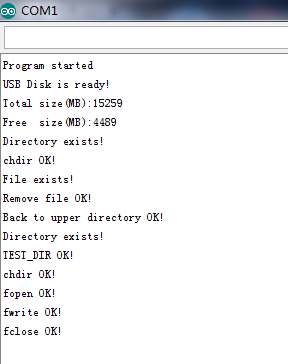
④ボードのUART0（FNST SRAM拡張ボードでもよい）をPCに接続し、Arduino IDEのSerial Monitorを開く。ボーレードを9600に設定する。



⑤USB DiskをHOSTインタフェースに挿入する。



⑥Serial Monitorウインドウで運行情報を参照する。



**(3)** MouseController とKeyboardController例の検証方法は上記と同様。