ECEN 5013

EMBEDDED SOFTWARE ESSENTIALS

PROFILING ON FRDM BOARD

Part 1 & 2 DMA and PROFILER

DMA MEMOVE CLOCK CYCLES

	10 byte	100 byte	1000 bytes	5000 bytes
1 byte	92	272	2074	10072
2 byte	86	175	1075	5075
4 byte	85	123	573	2572

DMA MEMOVE TIME IN MICROSECONDS

	10 byte	100 byte	1000 bytes	5000 bytes
1 byte	4	12	98	479
2 byte	4	8	51	241
4 byte	11	5	27	122

DMA MEMZERO CLOCK CYCLES

	10 bytes	100 bytes	1000 bytes	5000bytes
1 byte	93	274	2072	10071
2 byte	90	225	1574	7573
4 byte	85	176	1075	5076

DMA MEMZERO TIME IN MICROSECONDS

	10 bytes	100 bytes	1000 bytes	5000bytes
1 byte	4	13	98	479
2 byte	4	10	74	360
4 byte	11	8	51	241

STANDARD LIBRARY AND NON DMA FUNCTIONS ON FRDM BOARD

STANDARD LIBRARY MEMZERO ON FRDM

	10 BYTES	100 BYTES	1000 BYTES	5000 BYTES
TIME	7	37	152	1671

	10 BYTES	100 BYTES	1000 BYTES	5000 BYTES
CLOCK CYCLES	160	790	7101	35096

STANDARD LIBRARY MEMOVE ON FRDM

	10 BYTES	100 BYTES	1000 BYTES	5000 BYTES
TIME	4	48	434	2158

	10 BYTES	100 BYTES	1000 BYTES	5000 BYTES
CLOCK CYCLES	96	1018	9123	45117

Non DMA MEMZERO ON FRDM

	10 BYTES	100 BYTES	1000 BYTES	5000 BYTES
TIME	20	149	1149	5721

CLOCK CYCLES	10 BYTES	100 BYTES	1000 BYTES	5000 BYTES
CLOCK CYCLES	432	3132	24139	120146

NON DMA MEMMOVE ON FRDM

	10 BYTES	100 BYTES	1000 BYTES	5000 BYTES
TIME	29	256	1627	7997

SCREEN SHOTS

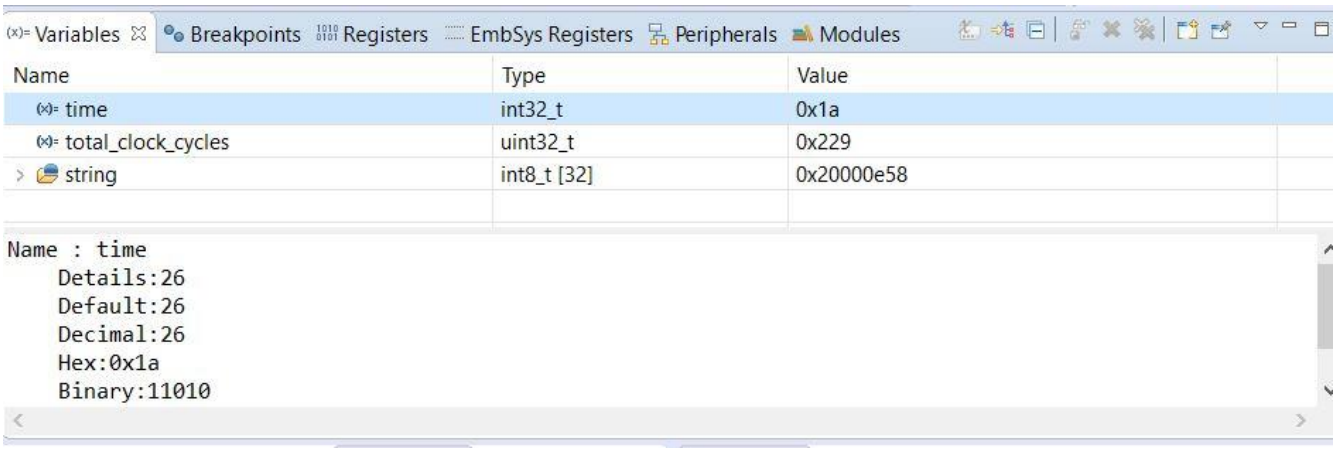

Fig 1 : The above figure shows time and clock cycles for transfer of 100 using 16 bit transfer memove.

Fig 2 : The above figure shows time ASCII value output on UART for 1000 byte transfer.

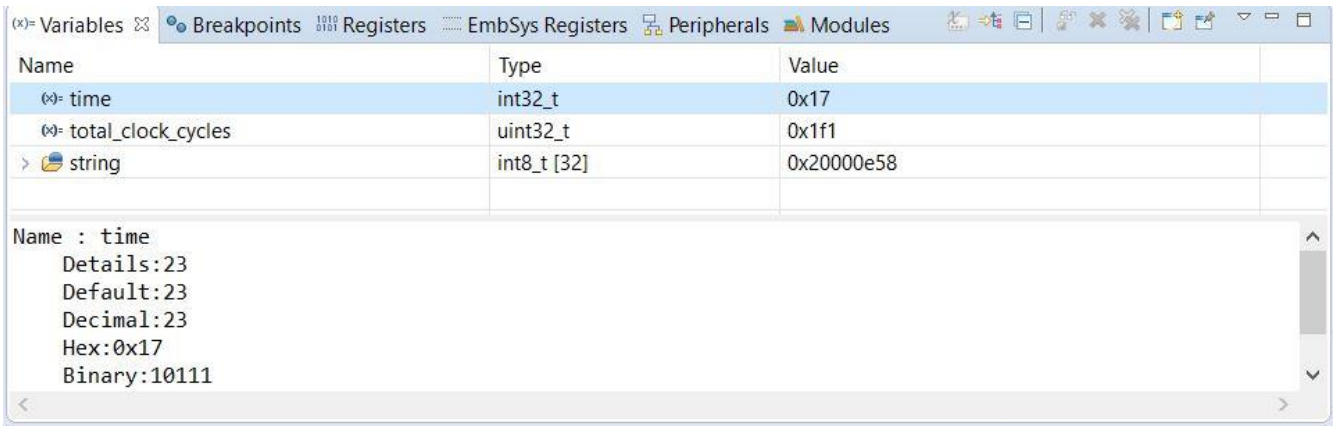

Fig 3 : The above figure show time and clock cycles for transfer of 100 bytes using 32 bit transfer memove.

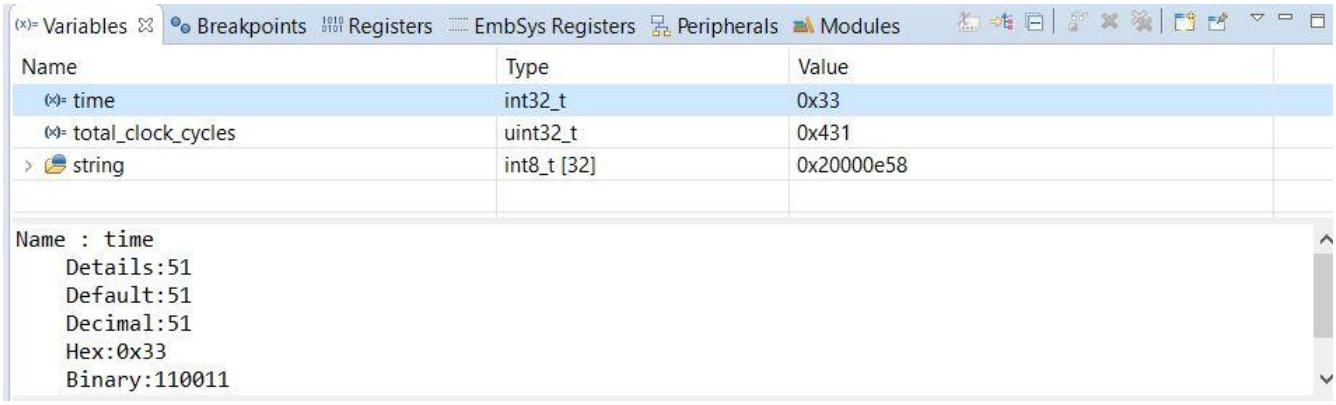

Fig 4 : The above figure show time and clock cycles for transfer of 1000 bytes using 16 bit transfer memove.

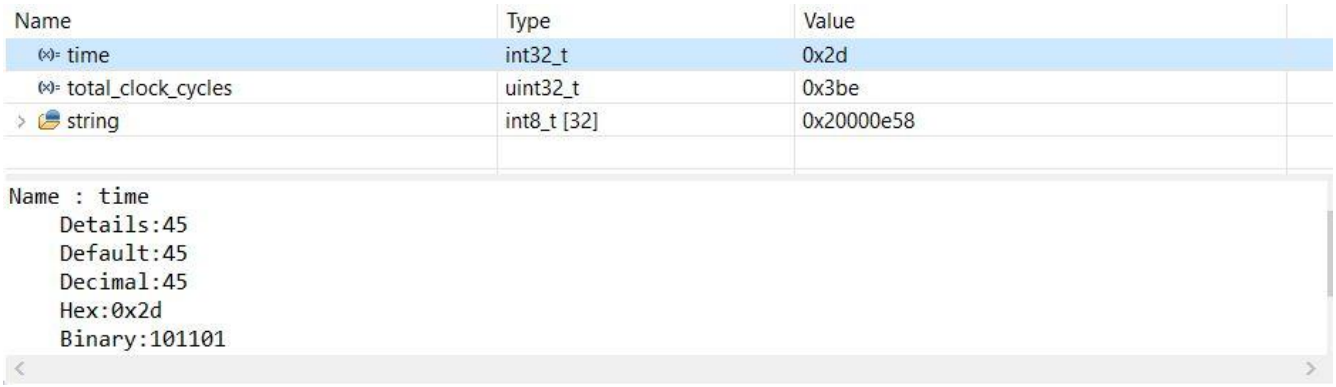

Fig 5 : The above figure show time and clock cycles for transfer of 1000 bytes using 32 bit transfer memove.

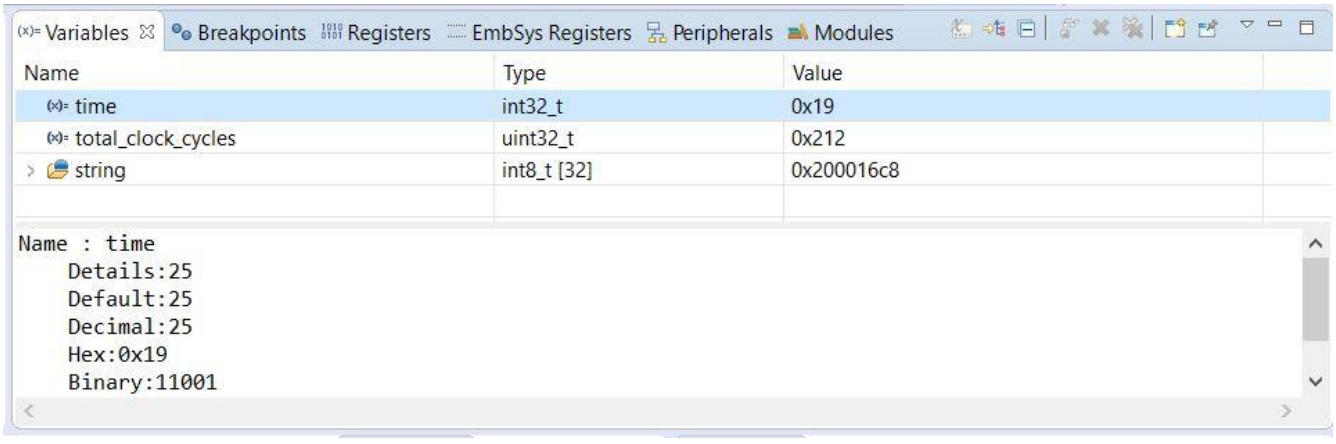

Fig 6 : The above figure show time and clock cycles for transfer of 100 bytes using 32 bit transfer memzero.

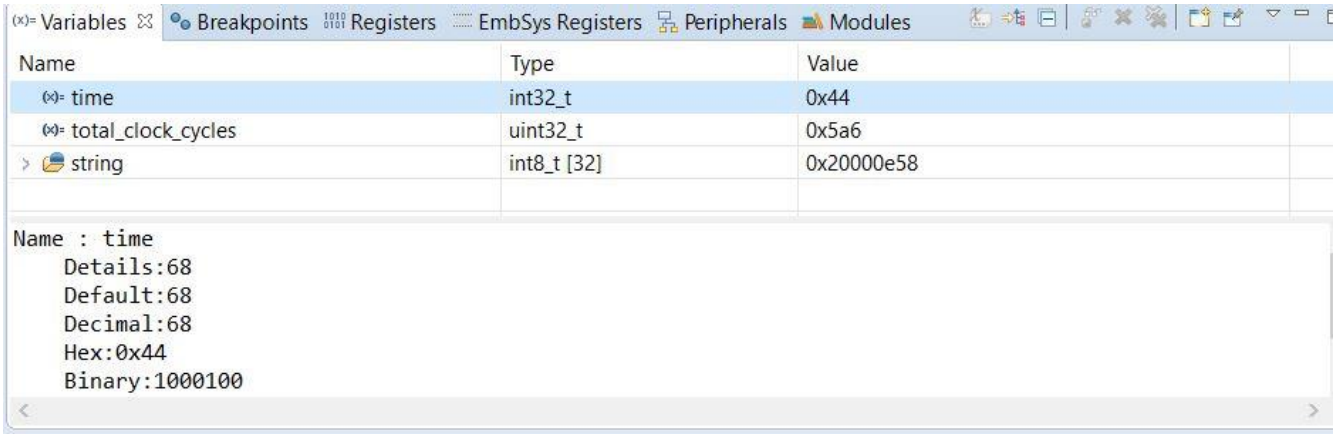

Fig 7 : The above figure show time and clock cycles for transfer of 1000 bytes using 32 bit transfer memzero.

PROFILING ON BBB BOARD

```
root@beaglebone:/home/debian/bin/project_3_BBB# vim memory.c
root@beaglebone:/home/debian/bin/project_3_BBB# gcc profiler_BBB.c
root@beaglebone:/home/debian/bin/project_3_BBB# ./a.out
Standard Library MEMSET PROFILING
 Time spent for 10 bytes for std libraray memset in seconds : 0.000005
Time spent for 100 bytes for std libraray memset in seconds : 0.000004
Time spent for 1000 bytes for std libraray memset in seconds : 0.000005
Time spent for 5000 bytes for std libraray memset in seconds : 0.000012
Standard Library MEMMOVE PROFILING
 Time spent for 10 bytes for std library memmove in seconds : 0.000025
Time spent for 100 bytes for std library memove in seconds : 0.000004
Time spent for 1000 bytes for std library memove in seconds : 0.000007
Time spent for 5000 bytes for std library memove in seconds : 0.000027
 NON DMA MEMMOVE PROFILING
 Time spent for 10 bytes of NON DMA memmove in seconds : 0.000007
Time spent for 100 bytes of non DMA memmove in seconds : 0.000015
Time spent for 1000 bytes of non DMA memmove in seconds : 0.000105
Time spent for 5000 bytes of non DMA memmove in seconds : 0.000576
 NON DMA MEMSET PROFILING
 Time spent for 10 bytes of non DMA memset in seconds : 0.000006
Time spent for 100 bytes of non DMA memset in seconds : 0.000011
Time spent for 1000 bytes of non DMA memset in seconds : 0.000077
Time spent for 5000 bytes  of non DMA memset in seconds : 0.000371
root@beaglebone:/home/debian/bin/project_3_BBB#
```

Fig 8 : The above figure shows profiling of Non DMA and standard library memove/memzero functions in seconds.

Part 3 - Logger and Buffer Enhancements

FRDM

Payload: 32-bit RTC_TSR Register value

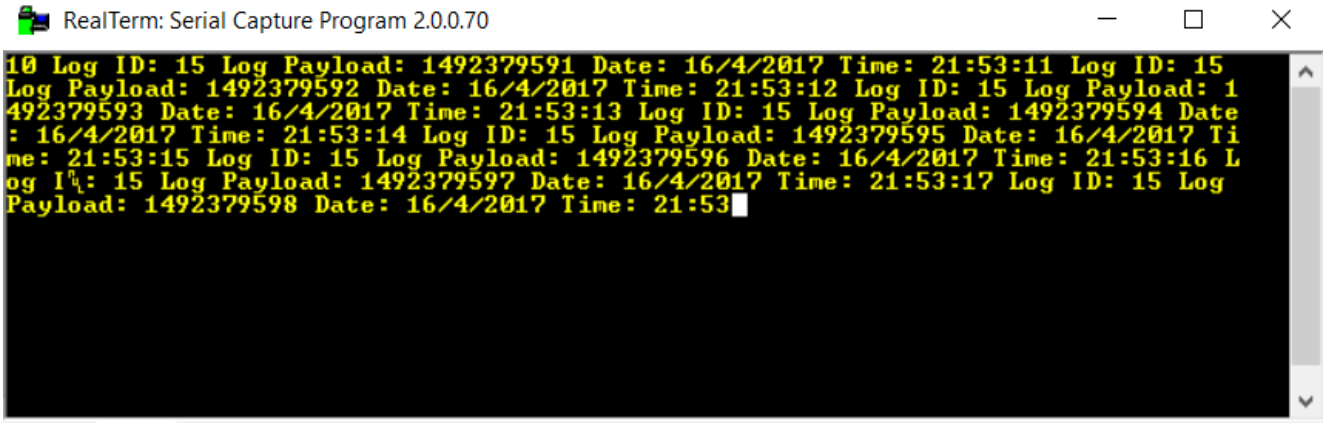

Fig 9: RTC Timestamp on FRDM KL25Z

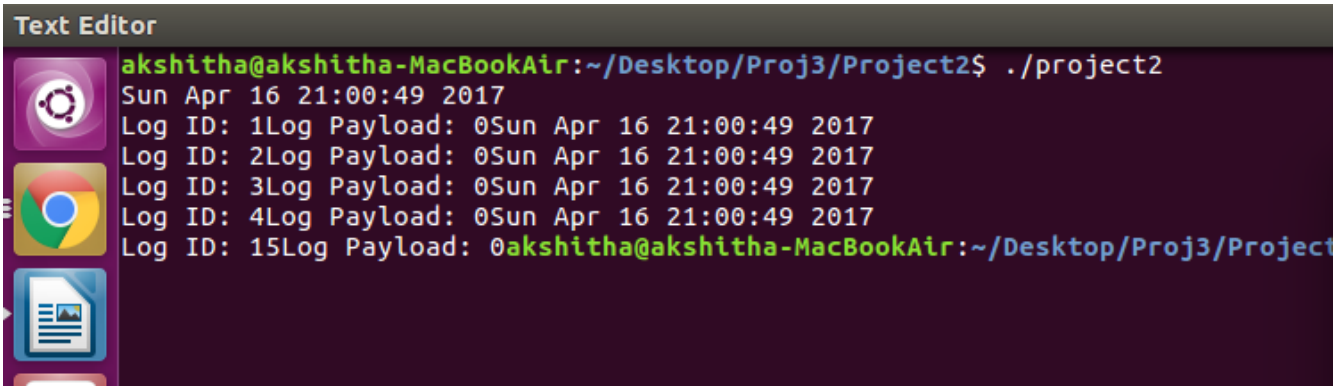

Fig 10: Timestamp using time.h library on Host Machine

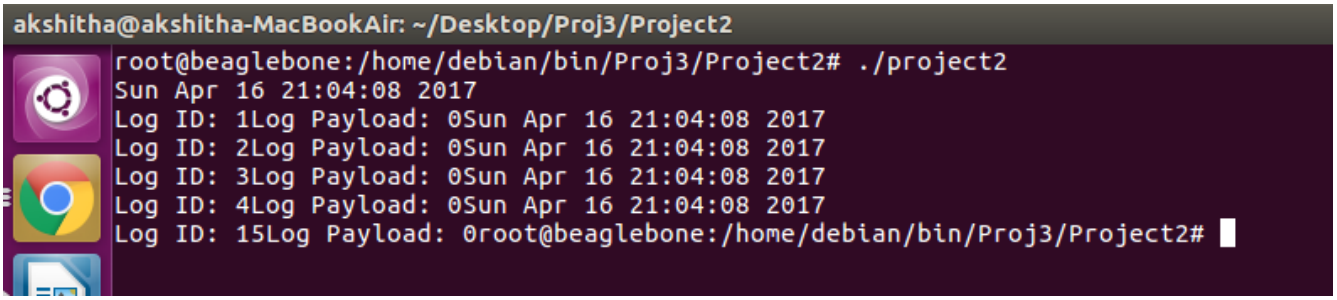

Fig 11: Timestamp using time.h library on Beagle Bone Black

Part 4 - Serial Peripheral Interface

The Nordic Registers were configured as follows:

Register	**Value Set**	**Value Read**
CONFIG register (read/write)	0x03	0x03
STATUS register (read)		0x0E
TX_ADDR (read/write)	0xA,0xB,0xC,0xD,0xE	0xA,0xB,0xC,0xD,0xE
RF_SETUP register (read/write)	0x02	0x02
RF_ch register (read/write)	0x05	0x05
FIFO_STATUS register (Read)		0x11

Screenshots:

core_cm0plus.h | DMA.c | gpio.h | gpio.c | main() at ma... | rtc.c | rtc.h | MKL25Z4.h | startup_MKL...

```
int main(void)
{
    uint8_t config_val, status_val, *tx_addr_val, rf_setup_val, rf_ch_val, fifo_status_val, tx_addr[5];

    SPI_init();

    while(1){

        nrf_write_config(NRF_CONFIG_PRIM_RX | NRF_CONFIG_PWR_UP);
        config_val = nrf_read_config();
        status_val = nrf_read_status();
        nrf_write_rf_setup(0x02);
        rf_setup_val = nrf_read_rf_setup();
        nrf_write_rf_ch(0x05);
        rf_ch_val = nrf_read_rf_ch();
        fifo_status_val = nrf_read_fifo_status();
        nrf_write_TX_ADDR();
        tx_addr_val = nrf_read_TX_ADDR();

        for(uint8_t i=0; i<5; i++)
        {
            tx_addr[i] = *tx_addr_val;
            tx_addr_val++;
        }
    }
    return 0;
}
```

Fig 12 : Writing and reading values to SPI

Variables | Breakpoints | Expressions | Registers | Peripherals | Modules

Name	Type	Value
config_val	uint8_t	0x3
status_val	uint8_t	0xe
tx_addr_val	uint8_t *	0x1ffff085
rf_setup_val	uint8_t	0x2
rf_ch_val	uint8_t	0x5
fifo_status_val	uint8_t	0x11
tx_addr	uint8_t [5]	0x20002fe0
tx_addr[0]	uint8_t	0xa
tx_addr[1]	uint8_t	0xb
tx_addr[2]	uint8_t	0xc
tx_addr[3]	uint8_t	0xd
tx_addr[4]	uint8_t	0xe

Fig 13 : Values read from Nordic Registers.