A first look at the Mitsubishi M16C 16-bit microcontroller

An independent review by Richard Sikora, Design Engineer.

When microcontrollers are mentioned, for many Mitsubishi is not a name that would immediately spring to mind. To raise the profile of the M16C family of 16-bit microcontrollers, a considerable amount of Mitsubishi advertising has recently appeared showing two aliens - or is it the management's perception of the electronics department? The advertising makes the following claims of the M16C family:

- A new architecture that combines the benefits of both accumulator and register based machines.
- Code execution performed in fewer cycles and uses less ROM.
- Facility to correct bugs, even after masking.
- All peripherals neatly stowed on board, so there is no need for external devices.
- CRC (cyclic redundancy check) calculation circuit for checksums.
- High levels of EMC and ESD protection.

Some of these claims are investigated later in this article.

A brief comparsion

The M16C range of microcontrollers was originally developed for Mitsubishi mobile telephones. The architecture bears close resemblance to that of the Hitachi H8 family, with which readers may already be familiar.

Table 1 below gives a brief overview of M16C facilities and compares them with one of Hitachi's best selling family of devices, the H8/300H. It can be seen from Table 1 that there are commonalties between the H8300H and the M16C. However, the M16C family does offer some facilities not normally seen on other families of microcontroller, namely:

- Choice of two clock sources, one for normal (fast) operation and the other for slow speed (low power) operation
- Selectable fundamental clock division ratio
- Choice of multiplexed or non-multiplexed address bus
- Three serial ports on the M16C61

The H8300H has stepper motor outputs and 16MB of address space, neither of which are provided by the M16C family.

Architecture

Registers R0-R3, A0, A1 and FB are arranged in two identical banks. Rather than having the bottleneck of an accumulator, registers R0, R1, R2 and R3 are used as working registers. Registers A0 and A1 are used as pointers memory or for general purpose.

Registers R0 and R1 can be used as either two16-bit registers or four 8-bit registers. Two stack pointers are provided, the User Stack Pointer (USP) for normal program operation and an Interrupt Stack Pointer (ISP). The two stack pointers are useful for implementing real-time operating systems, especially when used with the special 'context' instructions.

Boolean variables can be stored in terms of data type 'bit'. Bits can be useful for storing flags and can be used to implement encryption algorithms, etc. Addressing an individual bit is very simple indeed. State a starting address in memory and then add a number for the bit in the range 0-65535. The M16C offers eight timers but unlike the H8 does not have dedicated PWM registers. Instead, timers can be used to give 8-bit or 16-bit pulse width modulation, but it is not quite as straightforward as on the H8.

Instruction	Description
ADCF #immediate(byte and word)	Add with carry flag
ADD:Q # -8 to +7 (byte and word)	Add immediate to restricted range value
ADD #immediate (byte)	Add immediate to byte
AND #immediate (byte and word)	AND register with immediate value
CMP:Q # -8 to +7 (byte and word)	Compare register with restricted range value
CMP #immediate (byte)	Compare register with immediate value
DEC (byte and word)	Decrement register
INC (byte and word)	Increment register
INTO	Interrupt on overflow
LDC (byte and word)	Load register into control register e.g. FLG, SP,ISP,USP etc.
MOV #immediate (byte)	Move immediate value to register
NEG (byte and word)	Two's complement register contents
NOP	No operation
NOT (byte and word)	Invert register contents
OR #immediate (byte)	Logical OR immediate with register
ROLC and RORC (byte and word)	Rotate left and rotate right through carry.
STC	Load register from control register
STNZ #immediate (byte)	Store on non-zero.
STZ #immediate(byte)	Store on zero.
SUB #immediate(byte)	Subtract immediate value from register

Table3: M16C instructions operating in one clock cycle

Instruction set

The M16C instruction set is very similar to that of the H8500 which in turn resembles the Motorola 68000. Instuction syntax is the same as Hitachi and Motorola, i.e. from left to right with instruction, source, destination - for example `mov R1,R2`

For the following operations, there are commonalties between 32 of the M16C, H8500 and H8300 instructions. In many cases the mnemonics are identical. Examples are listed in Table 2.

The M16C offers the following improvements over the H8300H and H8500:

- More bit operations than on H8500

Mathematical operations	ADD, SUB
Logical operations	AND, OR
Bit Operations	BCLR, BTST, BSET, BNOT
Compare	CMP
Rotates	ROR, ROL
Sub-routine calls	JSR
Sub-routine returns	RET

Table 2: Instructions common to M16C, H8300 and H8500

- Stack frames (not available on H8300H)
- Multiple register operations (not available on H8300H)

On the other hand, there are one or two operations that the H8300H and H8500 can do that are provided on the M16C:

- Bit accumulator on H8300H (T bit)
- Moves to peripheral in synchronisation with E clock (for 6800 style peripherals)
- More choice of operations on control register
- 32-bit operations on the H8300H

	Device		
	Hitachi H8/300H	**Mitsubishi M16C60**	**Mitsubishi M16C61**
Number of pins	100	100	100
Clock division ratio	1:1	Configurable 1:1, 2:1, 4:1, 8:1 or 16:1	Configurable 1:1, 2:1 4:1, 8:1 or 16:1
Maximum clock frequency	16 MHz	10 MHz	10 MHz
RAM (bytes)	512 - 4K	10K	4K - 10K
On board ROM	16K -128K	64K	64K
Port pins in single chip mode	78	87	87
Port pins in external mode	46 (H8/3002)	52	52
Serial Ports	2	2	3
Analogue to digital converter	8 channels/ 10 bits	8 channels/ 10 bits	8 channels/ 10 bits
ADC Conversion time	8.4mS	3.3mS	3.3mS
Digital to analogue converter	None on H8/3002 2 on H8/3042/H8/3048	8 bits 2 channels	8 bits 2 channels
Pulse width modulation	2 x 8 bit	5 x 16 bit or 8 bit using timer	5 x 16 bit or 8 bit using timer
Stepper motor outputs	16	No	No
16 bit timers	5 channels	5 timers + 3	5 timers + 3
Direct Memory Access Controller	4 channels	2 channels	2 channels
Watchdog	Yes	Yes	Yes
CRC generating circuit	No	Yes	Yes
Multiplexed Address Bus	No	Yes	Yes
Non-multiplexed address bus	Yes	Yes	Yes
Address Space	1 MB or 16 MB	1MB	1MB
External Interrupts	9	4	4
Chip select lines	4	4	4

Table 1: Comparison of features of Hitachi H8 and Mitsubishi M16C devices

Execution times

The execution times of M16C instructions varies from one cycle for simple byte operations to more than ten cycles for complex operations. By and large, operations done directly on registers take two cycles (18 per cent of all instructions) and operations on indirectly addressed values in RAM take three cycles (50 per cent of all instructions).

Unlike the H8 family, the M16C family does have a certain number of instructions which operate in a single cycle. Even though this only represents a small percentage (2.1 per cent) of all the instructions, they represent some of those most commonly used and are listed in Table 3:

Special instructions

The M16C offers some instructions that are unavailable on the H8300H and the H8500:

The ABS function puts the absolute value of a register into that register, e.g. if value in register R0L is 8 or -8, then after `ABS.B R0L` instruction, R0L will contain 8.

Operating system commands are supported by the LDCTX and STCTX instructions. The STCTX instruction allows up to eight registers (an operating system task or context) to be saved in one operation.The LDCTX instruction restores up to eight registers already stored (loads an operating system task). An unusual instruction is the move nibble instruction which can move a nibble from one register to another:

MOVHL R0L, R1L; Puts high nibble in R0L into low nibble of R1L

The RMPA instruction has applications in digital filtering. It is used to generate the sum of inputs multiplied by appropriate filter constants.

e.g. Result = a x constA + b x constB + c x constC etc

Simplifications to implementation high-level language constructs are the conditional instructions STZ, STNZ and STZX (for bytes) and BMCND (for bits). The following piece of code can be implemented to execute in three cycles:

```
unsigned char x, y;          /* Allocate x to R0L and y to R0H */
if ( x == 7 )                /* CMP #7,R0L */
 y = 5;
else  y = 6;                 /* STZX #5, #6, R0H */
```

Program correction

Instruction	Description
ABS	Calculates absolute value of variable
ADJNZ, SBJNZ	Add jump non-zero, subtract jump non-zero. Used for iteration loops.
BTSTC, BTSTS	Bit test clear, bit test and set
BMCND	Bit move conditional
LDCTX, STCTX	Load / store context
MOVLL, MOVHL, MOVLH, MOVHH	Operations on nibbles
RMPA	Repeat multiply and addition
SMOVB, SMOVF, SSTR	Move strings (or data blocks)
STZ, STNZ, STZX	Store conditional on zero, non-zero, zero with extension

Table 4: M16C instructions unavailable on Hitachi H8300 or H8500

One of the interesting facilities provided by the M16C family is the ability to make a maximum of two corrections to masked ROM. There may be more than one reason for doing this. A bug may appear after the mask has been finalised or it may be necessary to make a program change to sell the finished product in a new country.

Code correction makes use of two address match interrupts. When the program counter reaches one of two preset values, program flow is diverted away from the mask ROM and runs instead from the corrected (patch) code in RAM.

Each code correction can correct more than one bug if a common point before the bugs occur can be identified and intercepted. However, in order to make use of the code correction facility, both hardware and software must be designed in a specific way.

One simple hardware implementation is to put the patch code into an external serial EEPROM. Devices such as the 24CXX family are not expensive, come in various sizes and may already be used on the PCB for non-volatile storage. If there is not an on-board EEPROM, then room for a socket should be laid out on the PCB.

In software, during initialisation, the interrupt vector table must be copied from ROM to RAM. The patch code must also be copied from EEPROM into RAM. Under normal circumstances the two address match interrupts would be assigned to unreachable code and would, therefore, have no effect.

However, when a patch EEPROM is present, address match interrupt 0 and address match interrupt 1 are loaded from addresses in EEPROM . When these addresses are reached in program flow, the code diverts to the patch in RAM instead of the masked ROM.

There are two limitations to program correction. First, it is not possible to patch boot code. Secondly, the size of patch code is limited by the total amount of RAM in the system.

CRC generation

A CRC (cyclic redundancy check) checksum gives a much higher level of error checking than can a longitudinal parity check (XOR of bytes) or a simple checksum. The disadvantage of the CRC is usually execution time. Each character of the CRC can take about 80 clock cycles to be generated using conventional instructions.

The M16C can do a 16-bit CRC using the CCITT standard (X16 + X12 + X5 + 1) in two clock cycles using a hardware implementation. One use for a CRC is to test the integrity of the code in ROM at power-up. In the event of there being a corrupted cell, then the code will not be run. A second use for the CRC is serial communications.

Documentation

Documentation for the M16C family is by and large produced in Japan and, therefore, some of the translations could be improved. Usually, the meanings are still clear.

Some of the instructions are not clearly explained and give no example, e.g. the interrupt for undefined instruction UND. There is similarity in the naming and operations of the three microcontrollers; see Table 5 for more information

Support tools

An ANSI 'C' compiler is available from IAR and a development board with on-board debugger interfacing to a PC has be just been released. Two Mitsubishi emulators are available - the high specification emulator with real-time trace and a lower cost device without trace. Emulators interface via the 100 pin PLCC connector rather than the flat pack connector which can be problematic at the best of times. Also available is an unpopulated printed circuit board which is laid out for a 100 pin PLCC socket (provided), RS232 port, two crystals, MAX232, bread-board area, etc. This costs in the region of £50

Conclusion

The M16C family offers some features not found on other 16-bit microcontrollers - built in CRC checksum generator, facility to make changes to masked ROM and three serial ports on the M16C61.

The device also offers the user a wider range of choices than competitive products - adjustable clock division ratio, choice of multiplexed/non-multiplexed address buses and two crystal sources. In terms of instruction set, the M16C offers the best of both H8300 and H8500 families plus some extras. Overall, the Mitsubishi M16C family device looks to be a good alternative to the Hitachi H8.

Richard Sikora
Design Engineer

Reprinted from the June, 1997, issue of Micro Technology Europe & MDA.

	M16C	H8 500	H8 300
Add	ADD	ADD	ADD
And	AND	AND	AND
Bit clear	BCLR	BCLR	BCLR
Bit test	BTST	BTST	BTST
Bit set	BSET	BSET	BSET
Bit not	BNOT	BNOT	BNOT
Compare	CMP	CMP	CMP
Divide (unsigned)	DIVU	DIVXU	DIVXU
Jump unconditional	JMP	JMP	JMP
Jump indirect	JMPI	JMP@@	JMP@@
Jump conditional	JCnd	BCC	BCC
Jump sub-routine indirect	JSRI	JSR@@	JSR@@
Jump sub-routine	JSR	BSR/JSR	BSR/JSR
Load control register	LDC	LDC	LDC
Multiply unsigned	MULU	MULXU	MULXU
Two's complement	NEG	NEG	NEG
No operation	NOP	NOP	NOP
Invert all bits	NOT	NOT	NOT
Logical OR	OR	OR	OR
Move	MOV / MOVA	MOV	MOV
Return from interrupt	REIT	RTE	RTE
Rotate left with carry	ROLC	ROTXL	ROTXL
Rotate right with carry	RORC	ROTXR	ROTXR
Rotate ROTL/ROTR	ROT	ROTL/ROTR	
Return from subroutine	RTS	RTS	RTS
Subtract with borrow	SBB	SUBX	SUBX
Shift arithmentic	SHA	SHAL/SHAR	SHAL/SHAR
Shift logical	SHL	SHLL/SHLR	SHLL/SHLR
Store from control register	STC	STC	STC
Subtract without borrow	SUB	SUB	SUB
Wait (Sleep?)	WAIT	SLEEP	SLEEP
Exclusive OR	XOR	XOR	XOR

Table 5: Instructions that have very similar names/operations