MOTOROLA
intelligence everywhere™

digital dna™

Breakpoint (BKP) Module V1

Block User Guide

HCS12 Microcontrollers

S12BKPV1/D
Rev. 1.02
5/2003

MOTOROLA.COM/SEMICONDUCTORS

Revision History

Version Number	Revision Date	Effective Date	Author	Description of Changes
1.02	5/1/2003	5/1/2003	John Langan	Original release

Table of Contents

List of Figures

List of Tables

Section 1 Introduction: Breakpoint (BKP) Module

Section 2 External Signal Description

Section 3 Memory Map/Register Definition

Section 4 Functional Description

Section 1 Introduction: Breakpoint (BKP) Module

This section describes the functionality of the Breakpoint (BKP) sub-block of the HCS12 Core Platform.

A block diagram of the Breakpoint sub-block is shown in **Figure 1-1**. The Breakpoint contains three main sub-blocks: the Register Block, the Compare Block, and the Control Block. The Register Block consists of the eight registers that make up the Breakpoint register space. The Compare Block performs all required address and data signal comparisons. The Control Block generates the signals for the CPU for the tag high, tag low, force SWI, and force BDM functions. In addition, it generates the register read and write signals and the comparator block enable signals.

NOTE: *There is a two-cycle latency for address compares and for forces, a two-cycle latency for write data compares, and a three-cycle latency for read data compares.*

1.1 Overview

The Breakpoint sub-block of the Core Platform provides for hardware breakpoints that are used to debug software on the CPU by comparing actual address and data values to predetermine data in setup registers. A successful comparison will place the CPU in Background Debug Mode or initiate a software interrupt (SWI). The choice between Background Debug Mode and SWI is software selectable.

There are two types of breakpoints, forced and tagged. Forced breakpoints occur at the next instruction boundary if a match occurs and tagged breakpoints allow for breaking just before a specific instruction executes. Tagged breakpoints will only occur on addresses of program fetches. Tagging on data is not allowed; however, if this occurs nothing will happen within the BKP.

The range function of the BKP allows breaking within a 256-byte address range. The page function allows breaking within expanded memory. In data matching operations, 8-bit or 16-bit data can be matched. Forced breakpoints are mainly used on a read or a write cycle, but can be used on any bus cycle.

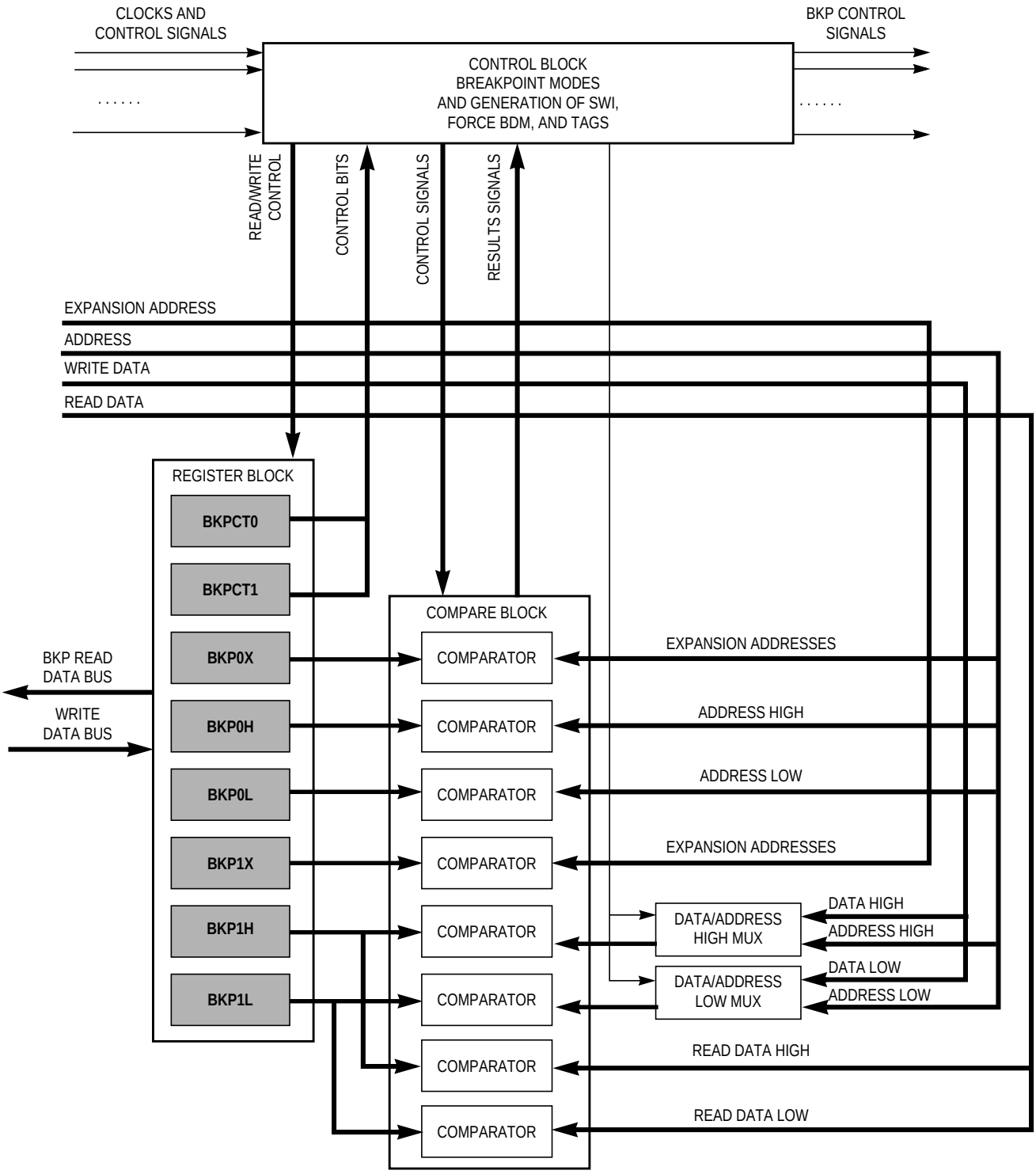

Figure 1-1 Breakpoint Block Diagram

1.2 Features

- Full or Dual Breakpoint Mode
  - Compare on address and data (Full)
  - Compare on either of two addresses (Dual)
- BDM or SWI Breakpoint
  - Enter BDM on breakpoint (BDM)
  - Execute SWI on breakpoint (SWI)
- Tagged or Forced Breakpoint
  - Break just before a specific instruction will begin execution (TAG)
  - Break on the first instruction boundary after a match occurs (Force)
- Single, Range, or Page address compares
  - Compare on address (Single)
  - Compare on address 256 byte (Range)
  - Compare on any 16K Page (Page)
- Compare address on read or write on forced breakpoints
- High and/or low byte data compares

1.3 Modes of Operation

The Breakpoint sub-block contains two modes of operation:

1. Dual Address Mode, where a match on either of two addresses will cause the system to enter Background Debug Mode or initiate a Software Interrupt (SWI).
2. Full Breakpoint Mode, where a match on address and data will cause the system to enter Background Debug Mode or initiate a Software Interrupt (SWI).

Section 2 External Signal Description

The breakpoint sub-module relies on the external bus interface (generally the MEBI) when the breakpoint is matching on the external bus.

The tag pins in **Table 2-1** (part of the MEBI) may also be a part of the breakpoint operation.

Table 2-1 External System Pins Associated With Breakpoint and MEBI

Pin Name	Pin Functions	Description
BKGD/MODC/ $\overline{\text{TAGHI}}$	$\overline{\text{TAGHI}}$	When instruction tagging is on, a 0 at the falling edge of PE4/ECLK tags the high half of the instruction word being read into the instruction queue.
PE3/$\overline{\text{LSTRB}}$/ $\overline{\text{TAGLO}}$	$\overline{\text{TAGLO}}$	In expanded wide mode or emulation narrow modes, when instruction tagging is on and low strobe is enabled, a 0 at the falling edge of PE4/ECLK tags the low half of the instruction word being read into the instruction queue.

Section 3 Memory Map/Register Definition

A summary of the registers associated with the Breakpoint sub-block is shown in **Figure 3-1**. Detailed descriptions of the registers and bits are given in the subsections that follow.

Address	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0028	BKPCT0	Read	BKEN	BKFULL	BKBDM	BKTAG	0	0	0	0
		Write								
$0029	BKPCT1	Read	BK0MBH	BK0MBL	BK1MBH	BK1MBL	BK0RWE	BK0RW	BK1RWE	BK1RW
		Write								
$002A	BKP0X	Read	0	0	BK0V5	BK0V4	BK0V3	BK0V2	BK0V1	BK0V0
		Write								
$002B	BKP0H	Read	Bit 15	14	13	12	11	10	9	Bit 8
		Write								
$002C	BKP0L	Read	Bit 7	6	5	4	3	2	1	Bit 0
		Write								
$002D	BKP1X	Read	0	0	BK1V5	BK1V4	BK1V3	BK1V2	BK1V1	BK1V0
		Write								
$002E	BKP1H	Read	Bit 15	14	13	12	11	10	9	Bit 8
		Write								
$002F	BKP1L	Read	Bit 7	6	5	4	3	2	1	Bit 0
		Write								

= Unimplemented X = Indeterminate

Figure 3-1 Breakpoint Register Summary

3.1 Breakpoint Control Register 0 (BKPCT0)

Read: anytime

Write: anytime

Register address $0028

	7	6	5	4	3	2	1	0
R	BKEN	BKFULL	BKBDM	BKTAG	0	0	0	0
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-2 Breakpoint Control Register 0 (BKPCT0)

This register is used to set the breakpoint modes.

BKEN — Breakpoint Enable

This bit enables the module
0 = Breakpoints disabled
1 = Breakpoints enabled, breakpoint mode is determined by bits BKFULL, BKBDM, and BKTAG

BKFULL — Full Breakpoint Mode Enable

This bit controls whether the breakpoint module is in Dual Mode or Full Mode
0 = Dual Address Mode enabled
1 = Full Breakpoint Mode enabled

BKBDM — Breakpoint Background Debug Mode Enable

This bit determines if the breakpoint causes the system to enter Background Debug Mode (BDM) or initiate a Software Interrupt (SWI)
0 = Go to Software Interrupt on a compare
1 = Go to BDM on a compare

BKTAG — Breakpoint on Tag

This bit controls whether the breakpoint will cause a break on the next instruction boundary (force) or on a match that will be an executable opcode (tagged). Non-executed opcodes cannot cause a tagged breakpoint
0 = On match, break at the next instruction boundary (force)
1 = On match, break if the match is an instruction that will be executed (tagged)

3.2 Breakpoint Control Register 1 (BKPCT1)

Read: anytime

Write: anytime

Register address $0029

	7	6	5	4	3	2	1	0
R W	BK0MBH	BK0MBL	BK1MBH	BK1MBL	BK0RWE	BK0RW	BK1RWE	BK1RW
Reset:	0	0	0	0	0	0	0	0

Figure 3-3 Breakpoint Control Register 1 (BKPCT1)

This register is used to configure the functionality of the Breakpoint sub-block within the Core.

BK0MBH:BK0MBL — Breakpoint Mask High Byte and Low Byte for First Address

In Dual or Full Mode, these bits may be used to mask (disable) the comparison of the high and low bytes of the first address breakpoint. The functionality is as given in **Table 3-1** below

Table 3-1 Breakpoint Mask Bits for First Address

BK0MBH:BK0MBL	Address Compare	BKP0X	BKP0H	BKP0L
x:0	Full address compare	Yes(1)	Yes	Yes
0:1	256 byte address range	Yes(1)	Yes	No
1:1	16K byte address range	Yes(1)	No	No

NOTES:
1. If page is selected.

The x:0 case is for a Full Address Compare. When a program page is selected, the full address compare will be based on bits for a 20-bit compare. The registers used for the compare are {BKP0X[5:0], BKP0H[5:0], BKP0L[7:0]}. When a program page is not selected, the full address compare will be based on bits for a 16-bit compare. The registers used for the compare are {BKP0H[7:0], BKP0L[7:0]}.

The 1:0 case is not sensible because it would ignore the high order address and compare the low order and expansion addresses. Logic forces this case to compare all address lines (effectively ignoring the BK0MBH control bit).

The 1:1 case is useful for triggering a breakpoint on any access to a particular expansion page. This only makes sense if a program page is being accessed so that the breakpoint trigger will occur only if BKP0X compares.

BK1MBH:BK1MBL — Breakpoint Mask High Byte and Low Byte of Data (Second Address)

In Dual Mode, these bits may be used to mask (disable) the comparison of the high and/or low bytes of the second address breakpoint. The functionality is as given in **Table 3-2**.

Table 3-2 Breakpoint Mask Bits for Second Address (Dual Mode)

BK1MBH:BK1MBL	Address Compare	BKP1X	BKP1H	BKP1L
x:0	Full address compare	Yes(1)	Yes	Yes
0:1	256 byte address range	Yes(1)	Yes	No
1:1	16K byte address range	Yes(1)	No	No

NOTES:
1. If page is selected.

The x:0 case is for a Full Address Compare. When a program page is selected, the full address compare will be based on bits for a 20-bit compare. The registers used for the compare are {BKP1X[5:0], BKP1H[5:0], BKP1L[7:0]}. When a program page is not selected, the full address compare will be based on bits for a 16-bit compare. The registers used for the compare are {BKP1H[7:0], BKP1L[7:0]}.

The 1:0 case is not sensible because it would ignore the high order address and compare the low order and expansion addresses. Logic forces this case to compare all address lines (effectively ignoring the BK1MBH control bit).

The 1:1 case is useful for triggering a breakpoint on any access to a particular expansion page. This only makes sense if a program page is being accessed so that the breakpoint trigger will occur only if BKP1X compares.

In Full Mode, these bits may be used to mask (disable) the comparison of the high and/or low bytes of the data breakpoint. The functionality is as given in **Table 3-3**.

Table 3-3 Breakpoint Mask Bits for Data Breakpoints (Full Mode)

BK1MBH:BK1MBL	Data Compare	BKP1X	BKP1H	BKP1L
0:0	High and low byte compare	No(1)	Yes	Yes
0:1	High byte	No(1)	Yes	No
1:0	Low byte	No(1)	No	Yes
1:1	No compare	No(1)	No	No

NOTES:
1. Expansion addresses for breakpoint 1 are not available in this mode.

BK0RWE — R/$\overline{W}$ Compare Enable

Enables the comparison of the R/$\overline{W}$ signal for first address breakpoint. This bit is not useful in tagged breakpoints.

0 = R/$\overline{W}$ is not used in the comparisons
1 = R/$\overline{W}$ is used in comparisons

BK0RW — R/$\overline{W}$ Compare Value

When BK0RWE = 1, this bit determines the type of bus cycle to match on first address breakpoint. When BK0RWE = 0, this bit has no effect.

0 = Write cycle will be matched
1 = Read cycle will be matched

BK1RWE — R/$\overline{W}$ Compare Enable

In Dual Mode, this bit enables the comparison of the R/$\overline{W}$ signal to further specify what causes a match for the second address breakpoint. This bit is not useful on tagged breakpoints or in Full Mode and is therefore a don't care.

0 = R/$\overline{W}$ is not used in comparisons
1 = R/$\overline{W}$ is used in comparisons

BK1RW — R/$\overline{W}$ Compare Value

When BK1RWE = 1, this bit determines the type of bus cycle to match on the second address breakpoint.When BK1RWE = 0, this bit has no effect.

0 = Write cycle will be matched
1 = Read cycle will be matched

3.3 Breakpoint First Address Expansion Register (BKP0X)

Read: anytime

Write: anytime

Register address $002A

	7	6	5	4	3	2	1	0
R	0	0	BK0V5	BK0V4	BK0V3	BK0V2	BK0V1	BK0V0
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-4 Breakpoint First Address Expansion Register (BKP0X)

This register contains the data to be matched against expansion address lines for the first address breakpoint when a page is selected.

BK0V[5:0] — Value of first breakpoint address to be matched in memory expansion space.

3.4 Breakpoint First Address High Byte Register (BKP0H)

Read: anytime

Write: anytime

Register address $002B

	7	6	5	4	3	2	1	0
R W	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Figure 3-5 Breakpoint First Address High Byte Register (BKP0H)

This register is used to set the breakpoint when compared against the high byte of the address.

3.5 Breakpoint First Address Low Byte Register (BKP0L)

Read: anytime

Write: anytime

Register address $002C

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 3-6 Breakpoint First Address Low Byte Register (BKP0L)

This register is used to set the breakpoint when compared against the low byte of the address.

3.6 Breakpoint Second Address Expansion Register (BKP1X)

Read: anytime

Write: anytime

Register address $002D

	7	6	5	4	3	2	1	0
R	0	0	BK1V5	BK1V4	BK1V3	BK1V2	BK1V1	BK1V0
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-7 Breakpoint Second Address Expansion Register (BKP1X)

In Dual Mode, this register contains the data to be matched against expansion address lines for the second address breakpoint when a page is selected. In Full Mode, this register is not used.

BK1V[5:0] — Value of first breakpoint address to be matched in memory expansion space.

3.7 Breakpoint Data (Second Address) High Byte Register (BKP1H)

Read: anytime

Write: anytime

Register address $002E

	7	6	5	4	3	2	1	0
R	Bit 15	14	13	12	11	10	9	Bit 8
W								
Reset:	0	0	0	0	0	0	0	0

Figure 3-8 Breakpoint Data High Byte Register (BKP1H)

In Dual Mode, this register is used to compare against the high order address lines. In Full Mode, this register is used to compare against the high order data lines.

3.8 Breakpoint Data (Second Address) Low Byte Register (BKP1L)

Read: anytime

Write: anytime

Register address $002F

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 3-9 Breakpoint Data Low Byte Register (BKP1L)

In Dual Mode, this register is used to compare against the low order address lines. In Full Mode, this register is used to compare against the low order data lines.

Section 4 Functional Description

The Breakpoint sub-block supports two modes of operation: Dual Address Mode and Full Breakpoint Mode. Within each of these modes, forced or tagged breakpoint types can be used. Forced breakpoints occur at the next instruction boundary if a match occurs and tagged breakpoints allow for breaking just before a specific instruction executes. The action taken upon a successful match can be to either place the CPU in Background Debug Mode or to initiate a software interrupt.

4.1 Modes of Operation

The Breakpoint can operate in Dual Address Mode or Full Breakpoint Mode. Each of these modes is discussed in the subsections below.

4.1.1 Dual Address Mode

When Dual Address Mode is enabled, two address breakpoints can be set. Each breakpoint can cause the system to enter Background Debug Mode or to initiate a software interrupt based upon the state of the BKBDM bit in the BKPCT0 Register being logic one or logic zero, respectively. BDM requests have a higher priority than SWI requests. No data breakpoints are allowed in this mode.

The BKTAG bit in the BKPCT0 register selects whether the breakpoint mode is forced or tagged. The BKxMBH:L bits in the BKPCT1 register select whether or not the breakpoint is matched exactly or is a range breakpoint. They also select whether the address is matched on the high byte, low byte, both bytes, and/or memory expansion. The BKxRW and BKxRWE bits in the BKPCT1 register select whether the type of bus cycle to match is a read, write, or both when performing forced breakpoints.

4.1.2 Full Breakpoint Mode

Full Breakpoint Mode requires a match on address and data for a breakpoint to occur. Upon a successful match, the system will enter Background Debug Mode or initiate a software interrupt based upon the state of the BKBDM bit in the BKPCT0 Register being logic one or logic zero, respectively. BDM requests have a higher priority than SWI requests. R/$\overline{W}$ matches are also allowed in this mode.

The BKTAG bit in the BKPCT0 register selects whether the breakpoint mode is forced or tagged. If the BKTAG bit is set in BKPCT0, then only address is matched, and data is ignored. The BK0MBH:L bits in the BKPCT1 register select whether or not the breakpoint is matched exactly, is a range breakpoint, or is in page space. The BK1MBH:L bits in the BKPCT1 register select whether the data is matched on the high byte, low byte, or both bytes. The BK0RW and BK0RWE bits in the BKPCT1 register select whether the type of bus cycle to match is a read or a write when performing forced breakpoints. BK1RW and BK1RWE bits in the BKPCT1 register are not used in Full Breakpoint Mode.

4.2 Breakpoint Priority

Breakpoint operation is first determined by the state of BDM. If BDM is already active, meaning the CPU is executing out of BDM firmware, Breakpoints are not allowed. In addition, while in BDM trace mode, tagging into BDM is not allowed. If BDM is not active, the Breakpoint will give priority to BDM requests over SWI requests. This condition applies to both forced and tagged breakpoints.

In all cases, BDM related breakpoints will have priority over those generated by the Breakpoint sub-block. This priority includes breakpoints enabled by the $\overline{\text{TAGLO}}$ and $\overline{\text{TAGHI}}$ external pins of the system that interface with the BDM directly and whose signal information passes through and is used by the Breakpoint sub-block.

NOTE: *BDM should not be entered from a breakpoint unless the BKEN bit is set in the BDM. Even if the ENABLE bit in the BDM is negated, the CPU actually executes the BDM firmware code. It checks the ENABLE and returns if enable is not set. If the BDM is not serviced by the monitor then the breakpoint would be re-asserted when the BDM returns to normal CPU flow.*

There is no hardware to enforce restriction of breakpoint operation if the BDM is not enabled.

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:
Motorola Literature Distribution
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

JAPAN:
Motorola Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu, Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

ASIA/PACIFIC:
Motorola Semiconductors H.K. Ltd.
Silicon Harbour Centre
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
852-26668334

HOME PAGE:
http://motorola.com/semiconductors

MOTOROLA

S12BKPV1/D
Rev. 1.02
5/2003