



Technical notes on using Analog Devices DSPs, processors and development tools
 Visit our Web resources <http://www.analog.com/ee-notes> and <http://www.analog.com/processors> or
 e-mail processor.support@analog.com or processor.tools.support@analog.com for technical support.

Host DMA Port on ADSP-BF52x and ADSP-BF54x Blackfin® Processors

Contributed by Gurudath Vasanth and Jayanti Addepalli

Rev 1 – May 6, 2008

Introduction

The Host DMA Port on ADSP-BF52x and ADSP-BF54x Blackfin® processors allows a host device external to the Blackfin processor to master DMA transactions in both directions. This EE-Note discusses details of using this port and provides example code for the different modes of operation. The Host DMA Port peripheral on ADSP-BF52x and ADSP-BF54x processors is functionally the same; differences have been summarized in [Appendix A](#).

Key Host DMA Port Features

- Allows an external host to configure and control DMA read/write data transfers and read port status.
- Uses a flexible asynchronous memory protocol
- 8/16-bit external data interface to host device
- Little/big endian data transfer
- Internal FIFO holds sixteen 32-bit words on ADSP-BF54x and sixteen 16-bit words on ADSP-BF52x processors
- Half-duplex communication
- Provides a mechanism for DMA bandwidth control

Host DMA Port Signals

Pin	Direction	Description
HOST_CE/	Input	Chip enable for Host DMA Port
HOST_WR/	Input	Write strobe
HOST_RD/	Input	Read strobe
HOST_ACK	Output	Flow control pin, termed as HRDY in acknowledge mode and FRDY in interrupt mode
HOST_ADDR	Input	Address pin 0: data port access 1: configuration port access
HOST_DATA <15:0>	Input/ Output	16-bit data port

Table 1. HOSTDP signals

HOSTDP Configuration

The Host DMA Port on the Blackfin processor must be configured by a host before data transfer. The host must poll the ALLOW_CONFIG bit in the HOST_STATUS register and ensure that this bit is set before initiating the configuration. The bit being set means that the slave is ready for configuration.

Seven 16-bit configuration words must be written in the following order.

1. HOST_CONFIG: describes the transfer flow (auto buffer mode or stop mode), DMA mode (1-D or 2-D), and DMA direction (read or write operation)

2. START_ADDR.L: Lower 16-bit start address – can be any valid internal or external memory
3. START_ADDR.H: Higher 16-bit start address – can be any valid internal or external memory
4. XCOUNT
5. XMODIFY
6. YCOUNT
7. YMODIFY

The amount of data moved between the host processor and the Host DMA Port (HOSTDP) should be a multiple of FIFO depth.

On ADSP-BF54x processors, the DMA bus of HOSTDP is 32 bits wide, but the data bus is either 8 or 16 bits wide. For consecutive memory reads/writes, the MODIFY value must be equal to 4.

Similarly, on ADSP-BF52x processors, the DMA bus of HOSTDP is 16 bits wide, but the data bus is either 8 or 16 bits wide. For consecutive memory reads/writes, the MODIFY

value must be equal to 2. The COUNT values also must be changed accordingly.

For example, for a 1-D write of 256 8-bit words to the HOSTDP on an ADSP-BF54x processor, XCOUNT would be equal to 64 and XMODIFY would be equal to 4.

Figure 1 and Figure 2 show logic analyzer plots of configuration for 16-bit host read transfers in acknowledge mode and 8-bit host write transfers in interrupt mode, respectively. If the HOSTDP is configured for 8-bit transfers, the host processor should send both bytes of a 16-bit configuration word.

The HOSTDP module does not forward descriptors to the DMA channel until it has received all seven configuration words. So, even if 1-D DMA mode is selected in HOST_CONFIG, it is necessary to send the YCOUNT and YMODIFY values (both 0 for 1-D DMA) to complete the HOSTDP configuration.

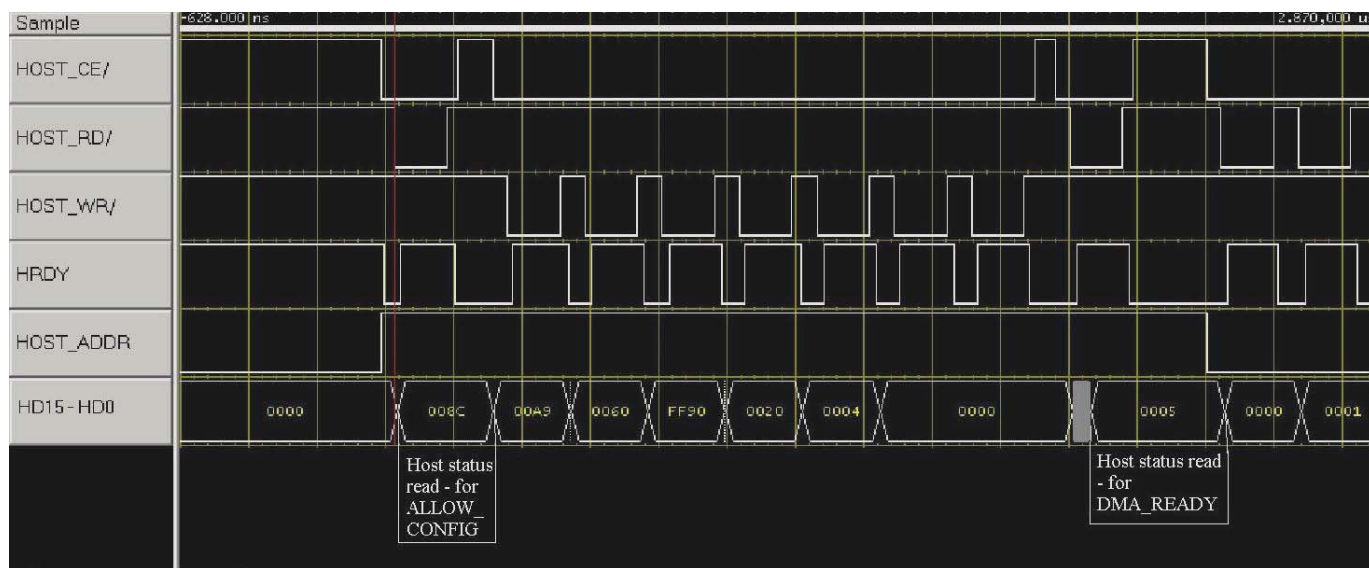


Figure 1. HOSTDP configuration for 16-bit host-read operation in acknowledge mode

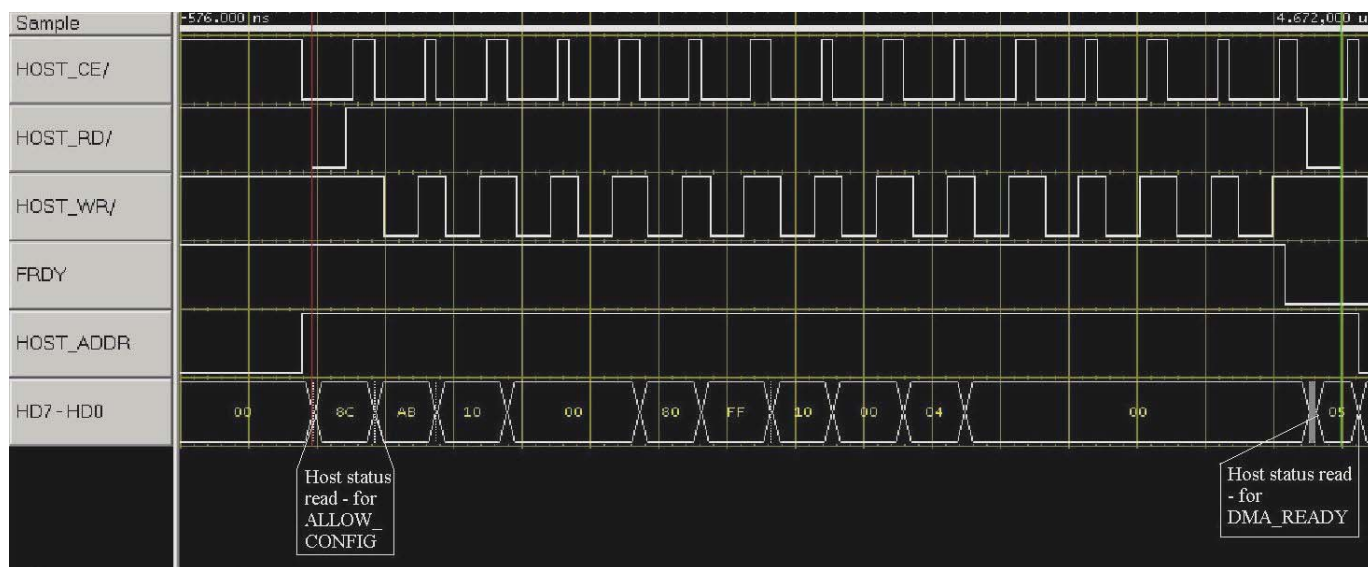


Figure 2. HOSTDP configuration for 8-bit host-write operation in interrupt mode

HOSTDP Transactions

There are four types of HOSTDP transactions. Each type of access is based on HOST_ADDR and

whether the HOST_RD or HOST_WR signal is asserted. The different transactions are tabulated in Table 2.

HOST_ADDR	/HOST_RD	/HOST_WR	/HOST_CE	Function
0	0	1	0	HOST read operation
0	1	0	0	HOST write operation
1	0	1	0	HOST_STATUS register read operation
1	1	0	0	HOST write configuration or control command

Table 2. HOSTDP transactions

HOSTDP Modes of Operation

There are two modes of flow control for the HOSTDP - interrupt mode and acknowledge mode. The HOST_ACK signal provides flow control between the host and the Blackfin processor, and is named differently, depending upon the mode of operation (Refer to Table 1). The flow control mode is configured by the slave processor when enabling the HOSTDP in the HOST_CONTROL register.

Interrupt Mode

In interrupt mode, a transition from high to low on the FRDY line indicates the FIFO status – FIFO empty in the case of host-write operations,

and FIFO full in the case of host-read operations. The host is expected to transmit/receive FIFO depth number of words on this transition.

Figure 3 shows an interface of HOSTDP with a host ADSP-BF537 processor. The corresponding code is given with the associated software for this EE-Note.

The Asynchronous Memory Interface of the ADSP-BF537 is connected to the HOSTDP. AMS3/ signal is connected to HOST_CE/. ARE/ is connected to HOST_RD/, and AWE/ is connected to HOST_WR/. FRDY is connected to a GPIO (PG13), which is configured to interrupt on a falling edge.

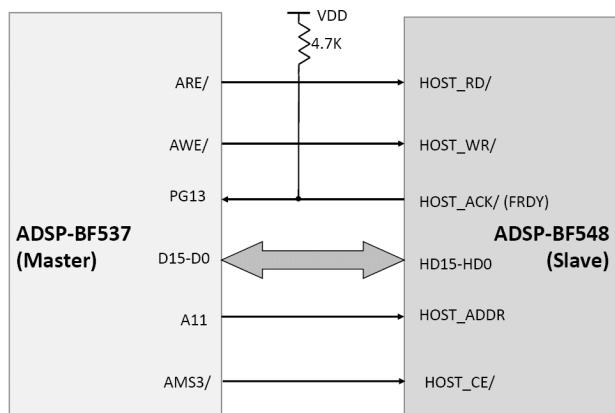


Figure 3. Interface of HOSTDP on an ADSP-BF548 processor with ADSP-BF537 host - interrupt mode

FIFO depth number of words are read/written from/to HOSTDP in the interrupt service routine. The reads/writes are done from/to address on Asynchronous Memory Bank 3, such that the address line (A11), which is connected to HOST_ADDR of the slave, is kept low during the data transfers. When writing configuration words or when reading HOSTDP status, this line must be maintained high.

Figure 4 shows a logic analyzer plot for 16-bit host-write operation, and Figure 5 shows the same for 16-bit host-read operation.

For host configuration writes or HOST_STATUS register reads, FRDY is not used for flow control.

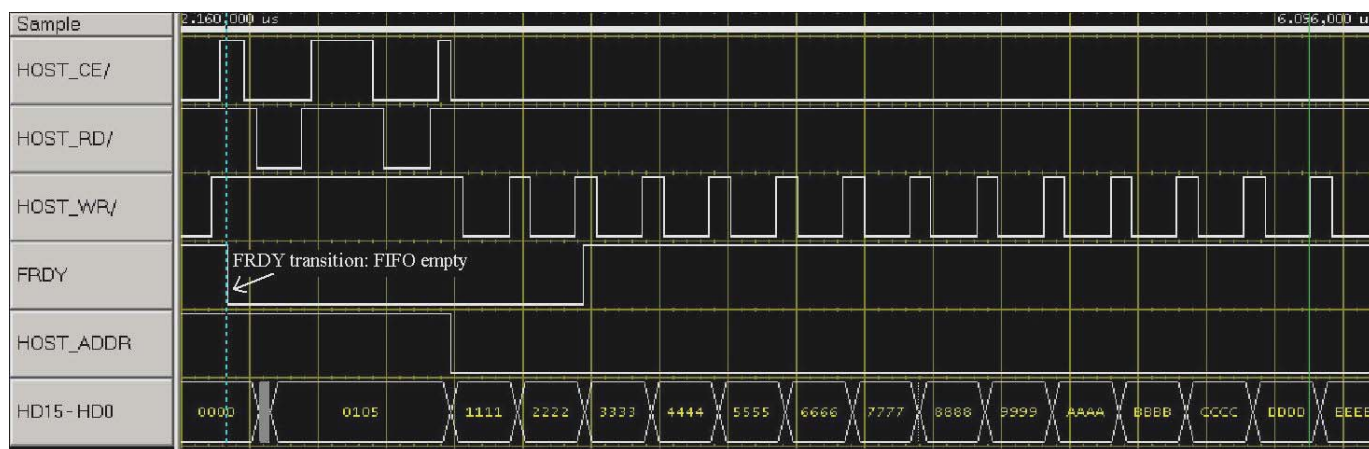


Figure 4. Host-write operation for 16-bit transfers in interrupt mode

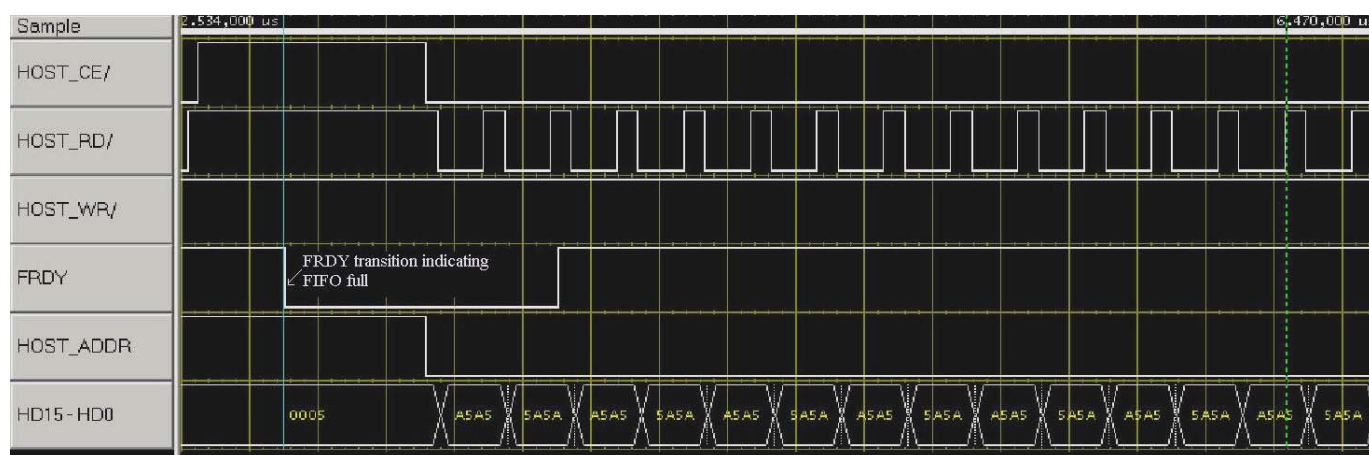


Figure 5. Host-read operation for 16-bit transfers in interrupt mode

Acknowledge Mode

In acknowledge mode, the HRDY signal is used to insert wait states to a read/write cycle. This mode is intended for use with host processors that support insertion of wait states through an external signal. On Blackfin processors, this signal is called ARDY. Figure 6 shows the interface of HOSTDP with an ADSP-BF537 processor.

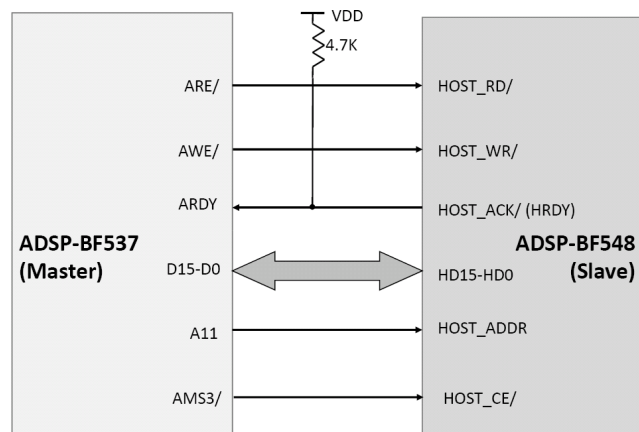


Figure 6. Interface of HOSTDP with ADSP-BF537 processor in acknowledge mode

The Asynchronous Memory Interface of ADSP-BF537 processor is connected to the HOSTDP. The AMS3/ signal is connected to HOST_CE/. ARE/ is connected to HOST_RD/, and AWE/ is connected to HOST_WR/. HRDY is connected to ARDY; a 4.7-Kohm pull-up resistor is used on this line. The reads/writes are done to any address of Asynchronous Memory Bank 3. The address is chosen in such a way that the address line (A11), which is connected to HOST_ADDR of the slave, is kept low during the data transfers. When writing configuration words or when reading HOSTDP status, this line must be maintained high.

Read/write accesses are done continuously by the host in this mode. When HOSTDP is unable to service read/write requests by the host because of the FIFO being empty/full, it inserts additional wait states by de-asserting HRDY. HRDY remains low for a period of $t_{DRDYRDL}$ for every read pulse and for a period of $t_{DRDYWRL}$ for every write pulse. Refer to the data sheet^[4] for timing specifications. If the FIFO is empty for a read access or full for a write access, HRDY remains de-asserted until the request is serviced.

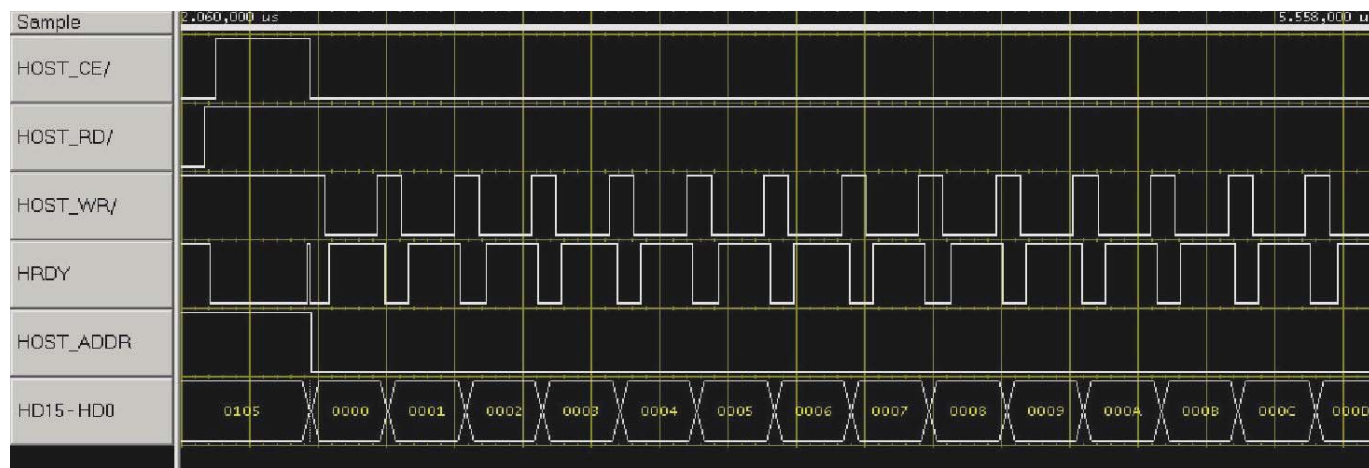


Figure 7. Host-write operation for 16-bit transfers in acknowledge mode

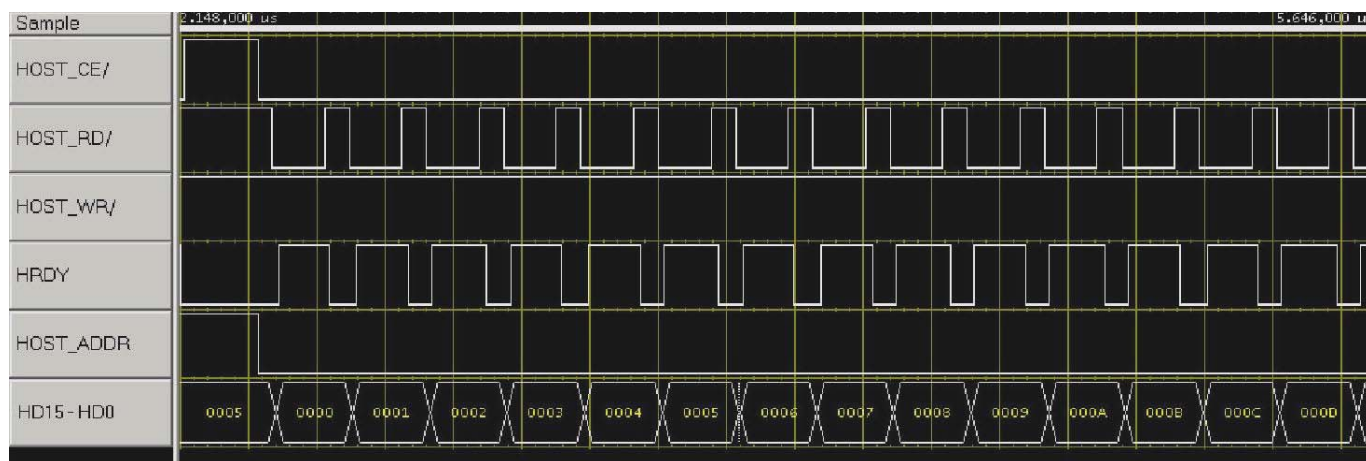


Figure 8. Host-read operation for 16-bit transfers in acknowledge mode

Control Commands

The host can send control commands to the Blackfin slave processor. Table 3 lists these commands.

The `HOST_IRQ` command provides a mechanism for the host to interrupt the `HOSTDTP` on the Blackfin processor. When the host writes this command, the `HIRQ` bit in the `HOST_STATUS` register is set and the `HOSTDTP` status interrupt is triggered.

The `DMA Finish` command provides a mechanism to terminate the `DMA` activity on the `HOSTDTP`.

HOST Data (HD7-HD0)	Command
b# 000111xx	<code>HOST_IRQ</code>
b# 001011xx	<code>DMA Finish</code>
b# 001111xx to 111111xx	Ignored

Table 3. Control commands

A control command is sent the same way as a configuration word (that is, by keeping the `HOST_ADDR` line high). When the `HOSTDTP` is waiting for configuration, control commands should not be sent. They would be misinterpreted as configuration words. It is recommended that the host ensures that the `ALLOW_CONFIG` bit in the `HOST_STATUS` register is cleared before issuing any command.

Appendix A

Table 4 lists differences between the Host DMA Port on `ADSP-BF52x` and `ADSP-BF54x` processors.

Parameters	ADSP-BF52x Processors	ADSP-BF54x Processors
FIFO size	16- x 16-bit words	16- x 32-bit words
DMA bus width	16 bits	32 bits
Interrupts	Separate interrupt for read and write operation. Bit 15 of the <code>HOST_STATUS</code> register describes this.	Common interrupt for both read and write operation.

Table 4. Differences in Host DMA Port on `ADSP-BF52x` and `ADSP-BF54x` processors

References

- [1] *ADSP-BF52x Blackfin Processor Hardware Reference*. Rev 0.3, September 2007. Analog Devices, Inc.
- [2] *ADSP-BF54x Blackfin Processor Hardware Reference*. Rev 0.1, March 2007. Analog Devices, Inc.
- [3] *ADSP-BF522/523/524/525/526/527 Blackfin Embedded Processor Preliminary Data Sheet*, Rev. PrD, December 2007, Analog Devices, Inc.
- [4] *ADSP-BF542/BF544/BF547/BF548/BF549 Blackfin Embedded Processor Preliminary Data Sheet*, Rev. PrG, December 2007. Analog Devices, Inc.

Document History

Revision	Description
<i>Rev 1 – May 6, 2008 by Gurudath Vasanth and Jayanti Addepalli</i>	Initial release